A QTE-based Solution to Keylogger Attacks

Chien-Wei Hung*, Fu-hau Hsu*, Shih-Jen Chen†, Chang-Kuo Tso*, Yan-Ling Hwang‡, Po-Ching Lin§ and Li-Pin Hsu¶
*Dept. of Comput. Sci. & Inform. Eng., National Central University, Zhongli City, Taiwan (R.O.C.)
Email: winestwinest@gmail.com; hsufh@csie.ncu.edu.tw; cktso@csie.ncu.edu.tw
†Institute for Information Industry, Taipei City, Taiwan (R.O.C.)
Email: jchen@iii.org.tw
‡Dept. of Applied Foreign Languages, Chung Shan Medical University, Taichung City, Taiwan (R.O.C.)
Email: yanling@csmu.edu.tw
§Dept. of Comput. and Inform. Sci., National Chung Cheng University, Chiayi County, Taiwan (R.O.C.)
Email: pclin@cs.ccu.edu.tw
¶Dept. of Applied Inform. Sci., Chung Shan Medical University, Taichung City, Taiwan (R.O.C.)
Email: apple@csmu.edu.tw

***Abstract*—Nowadays keystroke logging is one of the most widespread threats used for password theft. In this paper, rather than detecting existing malware or creating a trusted tunnel in the kernel, we present a different method QTE (Quick Time Events) to protect the password that a user provides for a web page to login to a web service. Installing such solutions in a host only requires limited privileges of related computers. The QTE method utilizes a canvas to cue users when their input will be recorded or ignored by the QTE add-on, which provides a chance for users to obfuscate keyloggers by inserting meaningless letters among the keystrokes of their passwords. QTE can be applied to all websites without any modification of them.**

***Keywords*-Authentication, computer security, keylogger, privacy.**

I. INTRODUCTION

Keystroke logging is a behavior of recording keystrokes, and a program with this property is called a keylogger. Typically, keyloggers sniff users' input, such as bank accounts or passwords, behind the scenes and upload the logs to attackers. Since keyloggers are easy to implement and produce high profit, about half of malware includes a keylogger in their code according to a Symantec's report [1]. In 2005, more than eight hackers were accused of planning to steal over 423 million dollars from a Japanese bank by installing keyloggers in her systems [2], which highlights the potential danger of keystroke logging.

In spite of many related works proposed various promising methods to defend keyloggers, they are less feasible in practice owing to the prerequisite of having root privileges or a dependable device. Besides, even though some companies like Google and Facebook provide the one-time password mechanism, it is only available for their services rather than all websites. Furthermore, in order to use their mechanism, one must reveal her/his personal data, such as a phone number, to each service provider and therefore sacrifices her/his privacy right indeed. To make matters worse, in last year, some reports [1][3] found that about 12% of leaked data were revealed by insiders and 96% of leaked information are personal data.

With the popularization of computers and Internet, it is very common for users to log in online banks or e-mail services using public computers. However, it is challenging to design a secure system in a public computer since common users only have limited privileges, and various persons can use the computer without being monitored. Therefore, an appropriate solution for keyloggers in this scenario must be able to be able to be actively executed by users without special privileges and applied to all websites. To meet the prerequisites, we propose the QTE method for users to defend their systems against keyloggers.

The QTE method provides a secure environment for users to input passwords in web browsers. The QTE method records a keystroke as a letter of a password only when the system is at a valid time interval. Valid time intervals are interleaved with invalid time intervals. Letters input at different time intervals are combined according to their time interval order. QTE utilizes animation to visualize valid time intervals, such as a special object moves over a specific screen area. QTE allows a user to obfuscate keyloggers by tapping keyboards haphazardly when the system is not at a valid time interval.

In this paper, we focus on practicable solutions when using public computers, and our work makes the following contributions:

1) This paper provides a feasible solution to obfuscate or bypass most kernel, hypervisor, hardware, and second-channel keyloggers even with a privilege-limited account in the user-space.
2) Our solution can be applied to all websites rather than a specific web service.
3) We do not rely on a password set beforehand or disclose users' information when using our approach.
4) Our solution will not slow down the speed of surfing the Internet.

 ISBN: 978-1-61208-209-7

5) Compared with most related work, our solution is easier to use but provide more powerful defense to keyloggers.

The rest of this paper is organized as follows. Section II discusses previous work about keyloggers with the evolution of attack and defense techniques. Section III depicts the principle and the system design of our solutions. Section IV gives the implementation details of our methods. Section V evaluates our solutions with seven different user-space and kernel-space keyloggers, and estimates the time an attacker has to consume in order to get a password under the protection of our mechanism. Experimental results show that our approach is immune to all tested keyloggers. Section VI concludes this paper and discusses the feasibility of extending our work to more applications.

II. Related Work

Since the first keylogger written by Perry S. Kivolowitz [4] appeared in 1983, keyloggers have become an essential element for most malware. While user-space keyloggers like Home Keylogger and Family Keylogger can be easily detected [5] and obfuscated by sending fake keystroke messages [6][7], most malware nowadays uses rootkit to hide itself in kernel or hypervisor. As a result, a lot of previous work was proposed and can be categorized into five types.

A. Signature-Based Solutions

A common way to detect kernel keyloggers is to use anti-keylogger software, such as SpyReveal. Anti-keylogger software detects keyloggers based on signatures [8]. For hardware keyloggers, Mihailowitsch suggested differentiating it according to the clock cycles of the keyboard [9]. However, both of them require known samples or devices for profiling; therefore, are resistless to new variants.

B. Encryption and Decryption

Another viewpoint adopted by both attackers and administrators utilizes cryptography to encrypt keystrokes in the device drivers and decrypts them in the applications [10]–[12]. To enhance the security, McCune used a mobile phone to communicate with the kernel module installed on the host with RSA encryption [13][14], and later the kernel module transmitted encrypted keystrokes to the registered application for decryption. Besides, similar principle is taken by Hirano who used a hypervisor-based isolation of trusted domain to exchange data over TLS/SSL [15].

These solutions provide stronger defense, but need root privileges of the related computers, which is inapplicable for public computer users, not to mention performance degradation caused by encryption and decryption. Moreover, with the evolution of hacking techniques, some keyloggers can even “upgrade” firmware for the purpose of concealment. A notable event is the exploitation of Mac OS X discovered by K. Chen [16]. More importantly, second-channel keyloggers are the focus of public attention nowadays. This kind of keyloggers can eavesdrop keystrokes via acoustics [17]–[19], electromagnetism [20], registers [21], power leakage, or even from optical sampling of mechanical energy [22]. In fact, solutions of this type are resistless to these keyloggers.

C. Graphical Password and On-Screen Keyboard

As keyloggers become more and more sophisticated, researchers continue developing new techniques to protect a computer against keyloggers. Graphical passwords [23]–[26] and on-screen keyboards [27][28], which make users type their passwords by mice, are new mechanisms extensively used by online banks to hedge against accounts theft.

Notwithstanding the new safeguards mentioned above, keyloggers also developed various approaches, such as taking screen snapshots and tracking cursor events, to bypass the protection or to facilitate information collection. The capability of screen capturing may be the most serious threat since everything displayed to users is also displayed to the attacker; the attacker can therefore speculate what users really type or click. In brief, an attacker can get passwords simply by a screen-shot and mouse event logs.

D. One-Time Password

One defensive way chosen by Facebook is one-time passwords [29]. Because each one-time password is only valid for one login session and is sent to a user through SMS by Fackbook, keystrokes logs become useless. Another variant of one-time passwords is the two-step verification [30] of Google that requires a random certificate as the second verification when logging in to a Google account.

One-time passwords are one of the strongest measures against keyloggers. However, users may need to render their real information, such as their cell phone numbers, to service providers, which may infringe users’ privacy surreptitiously. Furthermore, a critical defect is that a one-time password could only be recognized by its supporting website instead of all websites. In other words, if the website that a user tries to log in does not provide the one-time password mechanism, the user cannot be protected by the mechanism. Moreover, for each account on each website that provides one-time passwords, in order to enable one-time passwords, a user needs to make related setup. When a user has multiple accounts with one-time passwords, the management of the setup becomes a non-trivial work. However, our approach not only can be applied to all websites without their support, but also keep users’ privacy since they do not have to provide their private information to our mechanism.

E. Proxy Server

For the purpose of generality, Wu, Gieseke, and Pashalidis [31]–[34] recommend a proxy server as the intermedium between public computers and login servers. They keep a

 ISBN: 978-1-61208-209-7

copy of passwords on the proxy, and authenticate users with shared secrets defined beforehand such as personal questions or answer lists. These approaches provide a convenient and securer way to login to an account, but a compromised server or an unethical webmaster may seriously damage all related users. For this reason, Dinei [35] advices an adapted version that keeps nothing but password decryption algorithms used by users. However, this solution suffers non-trivial performance overhead.

III. System Design

Occasionally, a user has to type her/his password on a public computer. However, a public computer is usually a hostile environment since it may contain any software installed by any users, including skilled attackers. Besides, a normal user usually has only a limited privilege on a public computer. Thus, to develop a secure anti-keylogger solution on a public computer, the above limitations should be taken into account; otherwise, the proposed solution may not be a feasible one. To satisfy the above requirements, our solution assumes users have only limited privileges on public computers. Furthermore, our solution does not ask users to provide their private information or account information beforehand. Our solution is implemented as a browser extension that everyone can install using her/his account.

In our design, we assume there is no Man-In-the-Middle attack between a public computer with our extension and a website to which a user of the public computer tries to log in. But, if the website supports encryption, Man-In-the-Middle attacks can be solved by encrypting all messages. This section depicts the structures, components, advantages and disadvantages of the QTE method first. Implementation details will be expatiated in the next section.

A. QTE Method

QTE, which stands for Quick Time Event, is a video game term coined by Yu Suzuki [36]. Literally, QTE requires players to immediately perform some actions on control devices for on-screen prompts; otherwise, they will get some penalties or different outcomes. In other words, people may get nothing from the event if they do not react to the prompt in time. This idea is therefore be adopted by us to obfuscate most sorts of keyloggers.

1) Analysis: Most keyloggers have a full privilege of a compromised public computer while a user has a limited privilege in a public computer. Hence, forging any keystroke or hiding something by software may not be able to fool keyloggers. Specifically, the only thing that can interfere with most kernel, hypervisor, hardware, and second-channel keyloggers is what a user really types. As a result, we convert our problem to *"When could users type keys safely?" or "How could users know whether they could hit a keyboard haphazardly without affecting the original result?"*

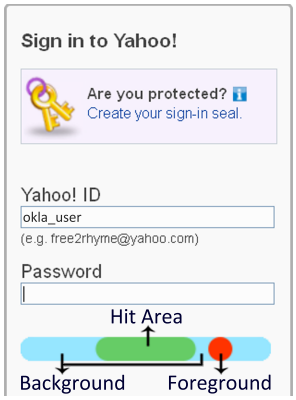

Figure 1. A view of canvas components.

2) Preliminary QTE Utilization: Remind that what users do will be ignored if they do not respond QTE in time. For this property, the QTE method adds a canvas to visualize both valid and invalid input areas on the screen. Furthermore, it divides the canvas into three parts: background, hit area, and moving foreground as shown in Figure 1. The hit area and background do not overlap with each other.

The most crucial parts of the QTE method are the hit area and the foreground. The foreground can be one or more moving objects that slide over the background. A user's input will be memorized by the QTE add-on extension as the characters of a password only if one moving object is over the hit area when the user presses the keyboard. The design therefore gives users a way to obfuscate most keyloggers by hitting their keyboards haphazardly when no foreground object is over the hit area. Besides, when the cursor is in a password filed, no matter what key a user presses on the keyboard, we replace it with an alphanumeric character and append it to the string in the password field to make keyloggers harder to retrieve the real password. The alphanumeric character is randomly generated. An example of this way is given in Figure 2, where we typed "`ABC`" when the moving foreground was over the background and "`123`" when it was over the hit area. Then the password field was stuffed with meaningless characters, "`i031Ta`", while the real password was stored by the QTE add-on somewhere in the heap segment. Afterwards, the real password was copied to the password field just before the form was submitted; therefore, users could still log in as though nothing was changed.

Despite the fact that the preliminary way seems sturdy enough to resist most keyloggers since users really press on keyboards; it is still vulnerable to those who can capture the screen whenever a user presses a key. To make the QTE method more secure, we revise the preliminary way slightly and advise an enhanced version.

3) Enhanced QTE Utilization: To begin with, we discuss functions about screen snapshots. So far as we know,

 ISBN: 978-1-61208-209-7

Figure 2. The result of typing ABC in background and 123 in hit area.

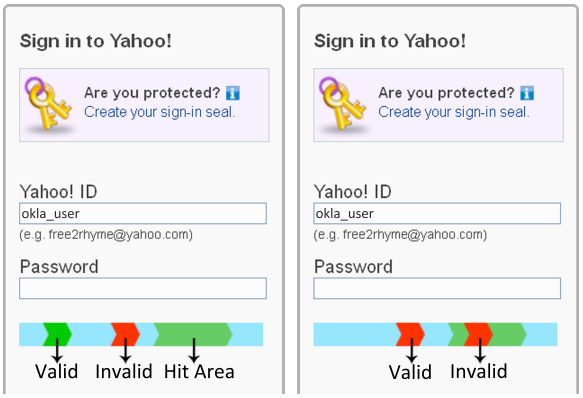

Figure 3. (a) The real entity was colored differently at the first second. (b) The real entity masqueraded itself since the next second.

commercial spyware takes a screen-shot in three situations according to periodical, event-triggered, and manual conditions; hence, it might be insufficient to take our first countermeasurement merely, if keyloggers are able to capture the screen. Nevertheless, screen capturing, which chose by the majority of spyware, has a great discrepancy with regard to screen recording: while the former simply gets a frame of a time point, the latter keeps track of every instant a user acts. Therefore screen capturing, in all probability, may miss to take a screen-shot at some critical moments, if they do not happen at the time specified a keylogger. Thus, if the events used to hint a user when she/he could type a valid character should be inferred from the continuous time, screen snapshots will no longer work.

Our prototype was inspired by the DDR machine, a musical game which requires users to tread on corresponding buttons when arrows on the screen become valid in a specific area. As the instances given in Figure 3, we randomly generate valid and invalid objects moving from left to right over the background and hit area. Keystrokes will be memorized as parts of a password only if a valid object is over the hit area. Both valid and invalid objects have the same shape. All valid objects have the same initial color, so do invalid objects. But the initial color of valid objects is different from the color of invalid objects. When approaching a hidden boundary, valid objects will disguise themselves as invalid objects by changing their color to the color of invalid objects. In this way, even keyloggers have the ability to log the screen whenever a user presses a key, they still cannot obtain a password.

By virtue of QTEs, we bring up two simple ways that are immune to most kernel, hypervisor, hardware, and second-channel keyloggers merely with normal privileges. While the first measure can not withstand those who can make snapshots, we reinforce our work and make an enhanced version that is resistant to them. Nonetheless, there might be few scenarios which demand the most secure and confidential procedure while screen recording is possible. With a view to apply for such circumstances, we address another innovative solution which works both safely and efficiently.

IV. Implementation

We implemented our solutions as a Firefox extension for the reason that every user can install the extension alone without special privileges of the related computer.

In order to visualize QTEs, we use the canvas element newly introduced in HTML5 standard [37]. Although the canvas element makes it easier to do animations in JavaScript, it costs more resources to maintain and repaint the same thing in different positions since the canvas element does not remember what it had drawn. As a result, we create only one canvas element per page to save performance overhead.

In practice, we firstly find all password fields when a webpage is loaded, and add event listeners for each password control. On this ground, we can easily insert or remove the canvas element when a control is focused or blurred. Secondly, we replace the default keyboard events of password fields with our method, which checks whether there exists at least one foreground object sliding over the hit area when detecting a keystroke. If the condition is true, we will store the key somewhere in the heap segment. However, when the cursor is in a password control, no matter what key a user presses on the keyboard, we replace it with a random alphanumeric character and append it to the string in the password control to simulate the original behavior. Eventually, when the form is being submitted, we restore the password control with the password string we have memorized.

V. Evaluation

Our evaluation is composed of two parts. Firstly, we examine the effectiveness of our solutions with seven different user-space or kernel-space keyloggers.

 ISBN: 978-1-61208-209-7

Table I
THE LIST OF KEYLOGGERS WE HAVE TESTED.

Keylogger	Level	Clip-board	Screen-shots	Packets Logging	Solve
Home	User	✓	✗	✗	✓
Family	User	✓	✗	Only URL	✓
Revealer	Kernel	✗	✗	✗	✓
Refog	Kernel	✓	✓	Only URL	✓
Ardamax	Kernel	✓	✓	Only URL	✓
All-In-One	Kernel	✓	✓	Only URL	✓
SpyAgent	Kernel	✓	✓	✓	✓

A. Experiments

To meet the real circumstances, we created a privilege-limited account on Windows XP SP3 in our machine. Besides, our system was installed with the Firefox web browser and seven different keyloggers including free and commercial trials. After that, we changed to the limited account and installed our extension without any special privileges. The list of keylogger features and experimental results are given in Table I.

At first, we tested the QTE method by typing the real password when a foreground was sliding over the hit area; otherwise, we just tapped on the keyboard arbitrarily. In the end, we found that the noise we produced were recorded by all keyloggers and made the real password indistinguishable. The experiment proved that keys we hit outside the hit area can successfully obfuscated attackers since the real password characters were interleaved with irregular strings. As a matter of fact, the outcome can be easily predicted since we really pressed the keys.

To sum up, our experiments showed the feasibility that one can bypass the monitor of keyloggers in web browsers even though the user has lower privileges than those of keyloggers'.

B. Attack Analysis

In this subsection, we analyze possible strategies that attackers can adopt to bypass QTE's protection.

If an attacker can obtain multiple keystroke samples which are used to input the same password under a QTE system, through common substrings analysis, the attacker can greatly increase his chance to get the real passwords. However, because a user usually will not use the same public computer frequently, an attacker may not be able to collect enough samples through a public computer. Besides, by adopting appropriate approaches to input a password, a user can increase the difficulty to leak the real password. What follows is an example. Assume `egi` is a password. A user can type a sequence of '`a`' first, then a sequence of '`b`', and then a sequence of '`c`', and so on, until he types a sequence of '`z`'. But only when '`e`', '`g`', and '`i`' are considered, the user will type them when the moving foreground is over the hit area. No other character will be typed when the moving foreground is over the hit area. As a result, no matter how many times the password is typed, a keylogger can only obtain a sequence of '`a`' followed by a sequence of '`b`' followed by a sequence of '`c`', and so on. The above information is not useful for the attacker. In the future, we plan to develop a new QTE component to execute the above operations automatically; thus, a user can input his password more quickly.

VI. CONCLUSIONS

While most solutions either require root privileges or leak users' information to service providers, we propose the QTE method to bypass most kernel, hypervisor, hardware, and second-channel keyloggers in the user-space even with a privilege-limited account. In general, the QTE method gives users a chance to obfuscate keyloggers by tapping keyboards haphazardly without affecting the original results. Moreover, we have tested seven different user-space and kernel-space keyloggers, and none of them could get our real passwords.

A. Contributions

Our work demonstrates the feasibility of bypassing almost all keyloggers, even users only have lower privileges, and it causes no latency after logging into the related services. Furthermore, the QTE method is compatible with all web-sites without the need of their support. To conclude, one can protect himself from keystroke logging with the QTE method. The QTE method can conquer nearly all keyloggers presumed the attacker will not record the screen and analyze them manually.

B. Future Work

In the future, one thing that we plan to finish is applying our solutions to all programs more than web browsers. For the moment, we can only catch edit controls of applications which use Win32 API to create user interface, and our approaches can be applied to them in the same way except that, instead of automatically inserting canvas or information tips next to password fields, users have to select the edit control on the screen manually. For other programs written in Java or some languages that draw edit controls in their ways, we can only make a screen-shot to ask users the position of edit controls, and simulate the mouse events to focus on it and fill passwords by user-space keystroke events. In most cases, since keyloggers do not want to be obfuscated by user-space events mentioned previously [6], [7], it may be a safer way to enter passwords in this manner at present.

 ISBN: 978-1-61208-209-7

ACKNOWLEDGMENT

Our work is funded by National Science Committee of Taiwan (ROC), and the numbers of the projects are NSC 99-2220-E-008-001 and NSC 99-2219-E-008-001.

REFERENCES

[1] Symantec, Symantec Internet Security Threat Report: Trends for 2010, vol. 16, Symantec, 2011.

[2] B. News, "UK police foil massive bank theft," BBC News, BBC News, 2005.

[3] InfoWatch, Global Data Leakage Report 2010, InfoWatch, 2011.

[4] P. S. Kivolowitz. "A Program To Allow ANYONE To Crack Unix (4.1 and 2)," http://securitydigest.org/unix/archive/006 06.11.2012.

[5] S. Ortolani, C. Giuffrida, and B. Crispo, "Bait your hook: a novel detection technique for keyloggers," Proc. the 13th International Conference on Recent Advances in Intrusion Detection, Ottawa, Ontario, Canada, 2010, pp. 198-217.

[6] Wuul, "AntiKeylogger," 2007.

[7] Wuul, "Log This!," 2008.

[8] SpyReveal, "SpyReveal," 2009.

[9] F. Mihailowitsch, Detecting Hardware Keylogger, 2010.

[10] F. J. Cini, Keystroke Encryption System, US, E. P. Kristina M. Grasso;Kristina M. Grasso, 2010.

[11] M. Kassner. "KeyScrambler: How keystroke encryption works to thwart keylogging threats," http://www.techrepublic.com/blog/security/keyscrambler-how-keystroke-encryption-works-to-thwart-keylogging-threats/4648 06.11.2012.

[12] A. Young and Moti Yung , "Deniable password snatching: on the possibility of evasive electronic espionage," Proc. IEEE Symposium on Security and Privacy, 1997 , pp. 224-235

[13] J. M. McCune, A. Perrig, and M. K. Reiter, "Bump in the ether: a framework for securing sensitive user input," Proc. the annual conference on USENIX '06 Annual Technical Conference. pp. 17-17.

[14] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978.

[15] M. Hirano, T. Umeda, T. Okuda, E. Kawai, and S. Yamaguchi, "T-PIM: Trusted Password Input Method against Data Stealing Malware," 2009 Sixth International Conference on Information Technology: New Generations, 2009, pp. 429-434.

[16] K. Chen, "Reversing and Exploiting an Apple®Firmware Update," in Black Hat, Las Vegas, Nevada, 2009.

[17] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," Proc. the 12th ACM conference on Computer and communications security, Alexandria, VA, USA, 2005, pp. 373-382.

[18] Y. Berger, A. Wool, and A. Yeredor, "Dictionary attacks using keyboard acoustic emanations," Proc. the 13th ACM conference on Computer and communications security, Alexandria, Virginia, USA, 2006, pp. 245-254.

[19] A. Kelly, "Cracking Passwords using Keyboard Acoustics and Language Modeling," School of Informatics, The University of Edinburgh, Edinburgh, 2010.

[20] M. Vuagnoux, and S. Pasini, "Compromising electromagnetic emanations of wired and wireless keyboards," Proc. the 18th conference on USENIX security symposium, Montreal, Canada, 2009, pp. 1-16.

[21] K. Zhang, and X. Wang, "Peeping tom in the neighborhood: keystroke eavesdropping on multi-user systems," Proc. the 18th conference on USENIX security symposium, Montreal, Canada, 2009, pp. 17-32.

[22] A. Barisani, and D. Bianco, "Sniff Keystrokes With Lasers/Voltmeters - Side Channel Attacks Using Optical Sampling Of Mechanical Energy And Power Line Leakage," in DEFCON 17, Riviera Hotel and Casino, 2009.

[23] M. N. Doja, and N. Kumar, "Image Authentication Schemes against Key-Logger Spyware," in ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. pp. 574-579.

[24] H. Wei, W. Xiaoping, and W. Guoheng, "The Security Analysis of Graphical Passwords," in International Conference on Communications and Intelligence Information Security. pp. 200-203.

[25] E. Stobert, A. Forget, S. Chiasson, P. C. v. Oorschot, and R. Biddle, "Exploring usability effects of increasing security in click-based graphical passwords," Proc. the 26th Annual Computer Security Applications Conference, Austin, Texas, 2010, pp. 79-88.

[26] A. P. Waterland, Secure password entry, NY, USA,to International Business Machines Corporation, 2009.

[27] K. Lab. "Kaspersky Internet Security 2009 includes a virtual keyboard that enables users to enter logins and passwords safely," http://www.kaspersky.com/news?id=207575675 06.11.2012.

[28] D. Hoover, Method and apparatus for secure entry of access codes in a computer environment, CA US Patent No. 6,209,102,to Arcot Systems, Inc., 2001.

[29] J. Brill, "More Ways to Stay Secure," More Ways to Stay Secure | Facebook, October 12, 2010, Facebook, 2010.

[30] N. Shah, "Advanced sign-in security for your Google account," Official Google Blog: Advanced sign-in security for your Google account, February 10, 2011, 2011.

[31] M. Wu, S. Garfinkel, and R. Miller, "Secure Web Authentication with Mobile Phones," in DIMACS Workshop on Usable Privacy and Security Software, 2004.

[32] A. Pashalidis, and C. J. Mitchell, "Impostor: a single sign-on system for use from untrusted devices," in IEEE GLOBECOM. pp. 2191-2195.

[33] E. Gieseke, and J. McLaughlin, "Secure Web Authentication with Mobile Phones Using Keyed Hash Authentication," Computer Science, Harvard University Extension, 2005.

[34] A. Pashalidis, "Accessing Password-Protected Resources without the Password," in WRI World Congress on Computer Science and Information Engineering. pp. 66-70.

[35] F. Dinei, and H. Cormac, "KLASSP: Entering Passwords on a Spyware Infected Machine Using a Shared-Secret Proxy," in Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual. pp. 67-76.

[36] T. Rogers, "Full Reactive Eyes Entertainment," Game Developer, December, 2010, 2010.

[37] A. C. Inc., M. Foundation, and O. S. ASA., "HTML Living Standard," Ian Hickson, 2011, pp. 288-315.

 ISBN: 978-1-61208-209-7