International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:14, No:6, 2020

Keyloggers Prevention with Time-Sensitive Obfuscation

Chien-Wei Hung, Fu-Hau Hsu, Chuan-Sheng Wang, Chia-Hao Lee

***Abstract*—**Nowadays, the abuse of keyloggers is one of the most widespread approaches to steal sensitive information. In this paper, we propose an On-Screen Prompts Approach to Keyloggers (OSPAK) and its analysis, which is installed in public computers. OSPAK utilizes a canvas to cue users when their keystrokes are going to be logged or ignored by OSPAK. This approach can protect computers against recoding sensitive inputs, which obfuscates keyloggers with letters inserted among users' keystrokes. It adds a canvas below each password field in a webpage and consists of three parts: two background areas, a hit area and a moving foreground object. Letters at different valid time intervals are combined in accordance with their time interval orders, and valid time intervals are interleaved with invalid time intervals. It utilizes animation to visualize valid time intervals and invalid time intervals, which can be integrated in a webpage as a browser extension. We have tested it against a series of known keyloggers and also performed a study with 95 users to evaluate how easily the tool is used. Experimental results made by volunteers show that OSPAK is a simple approach.

***Keywords*—**Authentication, computer security, keylogger, privacy, information leakage.

I. Introduction

SINCE keylogging is easy to be implemented and produces high profit, it becomes a common technique used by attackers. In fact, about one-half of malware include a keylogger in their code according to a Symantec's report [1]. In 2014, 16 million e-mail addresses and passwords have been hacked, according to German authorities. And, the keyloggers are thought to have been used in the theft [2].

Although many studies have proposed various promising methods to defend keyloggers [8]-[10], most of them require root privileges or a dependable device to install them, which are not feasible on public computers. Besides, even though some companies like Google and Facebook provide the one-time password mechanisms, the mechanisms are only available for their own services rather than all other websites, and they could only protect the passwords while other private data such as account name, e-mail content, credit card number, and personal identity number are not under the protection. With diverse services provided through the Internet, computers and networks have become a critical part of our everyday life; hence, once in a while, we may need to log in online to banks or e-mail services using public computers when we do not bring our laptops with us or when the screen of a smartphone is too small to use a website. However, it is a challenge to design a secure solution that can be installed in public computers because of the limited privileges of common users. In addition, malicious users may use a public computer without monitoring. Therefore, an appropriate solution in this scenario needs to be actively executed by users without special privileges and can be applied to every website. To satisfy these prerequisites, based on an earlier work, a QTE-based Solution [3], we conduct an extension of it and propose an approach called OSPAK. We conduct a series of experiments and analysis to defend the systems against keyloggers.

OSPAK provides a secure method for users to input sensitive information in most web browsers, even in a hostile environment, such as using a public computer. When the OSPAK is activated, it records a keystroke as a letter of an input string only in a valid time interval. Valid time intervals are interleaved with invalid time intervals. Besides, experimental results made by 95 volunteers also show that OSPAK is easy to learn for users, which is a simple approach.

II. Design

At times, a user has to type his password on a public computer. However, a public computer is usually a hostile environment. The public computer may contain some software installed by any user, including skilled attackers. Besides, a normal user usually only has a limited privilege on a public computer. OSPAK is implemented as a browser extension so that everyone can use it integrated in a webpage when logged in their private account. The main idea is to randomly divide time into two classes, one that accepts key strokes as passwords, and the other ignores key strokes. Visual cues are used to indicate the time division. Thus, it becomes difficult for attackers to steal any information from a mixture of valid password and obfuscated input [3].

OSPAK carries out a canvas, called OSPAK canvas, below each password field in a web page to visualize both valid and invalid time intervals to input password letters on the screen. An OSPAK canvas consists of three parts: two background areas, a hit area, and a moving foreground object. The hit area is inserted between the two background areas and does not overlap with them. The moving foreground object of an OSPAK canvas slides backwards and forwards over the background areas and hit area of the OSPAK canvas at a speed changing randomly. The valid time intervals of an OSPAK canvas are the time intervals when the moving foreground object of an OSPAK canvas is over the hit area of the canvas.

Fu-Hau Hsu is with National Central University and has had an appointment in the Department of Computer Science and Information Engineering, Taoyuan, 32001 Taiwan, ROC (e-mail: hsufh@csie.ncu.edu.tw).

Chia-Hao Lee is a PhD Candidate in the Department of Computer Science and Information Engineering, National Central University, Taoyuan, 32001 Taiwan, ROC (corresponding author, phone: 886-3-4227151 #35358; e-mail: diolee@csie.ncu.edu.tw).

The invalid time intervals of an OSPAK canvas are the time intervals when the moving foreground object of an OSPAK canvas is over a background area of the canvas. As the user types in the keyboard, an input letter will be memorized as a password character only if the foreground object is over the hit area when the user makes a keystroke. At first, all valid objects have the same color but once approaching a hidden boundary; valid objects will disguise themselves as invalid objects by changing their color to the color of invalid objects.

An input letter will be memorized by the OSPAK add-on extension as a constituent character of a password only if a foreground object is over the hit area when a user makes the corresponding keystroke. In other words, only letters input at a valid time interval will be recognized as constituent letters of a password. During each valid time interval, a user can input one or more constituent letters of his password. He can even idle without typing any letter in a valid time interval. When a user presses the submit button, letters input at previous valid time intervals are combined together according to the order of their corresponding valid time intervals to form the password that is going to be submitted to the related web site.

The proposed technique obfuscates information using time division. When the cursor is in a password field, no matter what key a user presses on the keyboard, OSPAK replaces it with inserted invalid key strokes and append it to the string in the password field, which is produced randomly. Here is an example of this way: a user can type "XYZ" when it is in a foreground input area of a webpage for a user login, and type “realpw” when it is over the hit area. Then, the password field will stuff with some meaningless characters, like "670Za4fF1", while the real password was stored by OSPAK somewhere in the heap segment. As a result, the real password will be copied to the password. Consequently, people’s privacy may be protected more safely by OSPAK when logging into a website [3].

III. Attack Analysis

In this section, we analyze possible strategies that attackers can adopt to bypass OSPAK's protection and how OSPAK can avoid such elusion. To evaluate the robustness of OSPAK under a hostile environment, we make various analyses in this section assuming that a user makes use of a public computer frequently; thus, almost all programs in the computer cannot be trusted and the previous input data of that user will be recorded. However, in the real world, this situation may not happen too often. Hence, some of our suggestions, such as downloading a browser, may not be convenient for a user. But the inconvenience occurs only when a user uses a public computer to input sensitive data. However, in the past, using a public computer to input sensitive data was very dangerous and should be avoided. However, with the help of OSPAK, it is now possible to do it safely.

The first problem OSPAK may face is that an attacker may replace a browser with his own modified version in a public computer. Under this situation, the browser will not execute downloaded OSPAK but execute fake OSPAK or it may steal passwords stored in the memory used by OSPAK. To avoid this problem, instead of using the browser in a public computer, a user can download a browser from the Internet first and then add OSPAK to it to surf the Internet. Nowadays, almost all major browser companies provide free browsers for users to download. Besides, even though after a clean browser is downloaded and executed, an attacker can use a plug-in to inject code into it, the plug-in cannot steal passwords stored in the memory of OSPAK. After all, the malicious plug-in cannot interrupt the execution of OSPAK. Moreover, after OSPAK finishes its job, it will clean its local memory used to store a password.

The second problem OSPAK may face is that an attacker may sniff the network traffic to intercept passwords, nevertheless, the password is normally transferred in HTTPS, and such encrypted mechanisms should be enough to protect against packet sniffing attack. It is also possible that attackers deceive users to visit fake website with a DNS spoofing attack. However, the problem is in the scope of phishing or DNS cache poisoning; hence, we do not discuss it in this paper.

The third problem OSPAK may face is as follows. If an attacker can obtain multiple keystroke samples which conform to the same password under an OSPAK system, through common subsequence analysis, the attacker can increase his chance to get the real password. However, by adopting appropriate approaches to input a password under OSPAK, a user can greatly increase the difficulty to obtain the real password. For example, assume the password of a user is `mfn3eb7o.` To help the user to remember what letters he needs to type so that all his input samples for the password have the same long common substring set, the user can associate each letter of his password with a word he is familiar with. A word could be the name of a person, the brand of a car company, the name of a university, and so on. Hence, the user can associate m with Amy, f with Sofia, n with Kevin, 3 and 7 with 0123456789, e with Jennifer, b with Robert and o with John. Then, whenever the user needs to type his password, to input the m in the password, the user can type A when the moving foreground object is over a background area, then he types m when the moving object is over the hit area, and finally he types y when the moving object is over a background area. As a result, only m will become a constituent letter of the user's password. But Amy becomes a common substring. The other constituent letters of the password are input in the same way. The above approach creates the following common string `"AmySofiaKevin0123456789JenniferRobert0123456789John"` with 51characters.

To further increase the length of a common subsequence of a password, in front of the word Amy, the user can type his grandfather's name when the foreground moving object is over a background area. And between the words Amy and Sofia, the user can type his grandmother's name when the foreground moving object is over a background area. And between the words Sofia and Kevin, the user can type his father's name when the foreground moving object is over a background area. Based on the above strategy, the user can easily create a common substring with 100 characters.

According to [4], the lengths of most passwords are between

6 characters and 12 characters. Therefore, after obtaining a 100-character substring, at the worst condition, a keylogger user needs to try $\sum_{i=6}^{12} C_i^{100} = 1.2*10^{15}$ times to find the correct password. Thus, at worst, $1.2*10^{15}$ TCP connections are needed to be established between the hosts controlled by the keylogger users and the related servers to test these $1.2*10^{15}$ possible passwords. Among these $1.2*10^{15}$ tries, only one try will succeed. The number shows that not only it is difficult to get the real password from the above common substring, but also before the keylogger user obtains the password, his hosts may already be detected; after all, most firewalls or IDSs can easily detect multiple login failures. Therefore, with the above difficulty, a user can even write the whole string he can use to input a valid password on a piece of paper without worrying that the paper may be stolen.

The game style password input method of OSPAK may be attractive to young people. However, inputting an extra 90-something characters to bypass the common substring problem may not be convenient for some people. Hence, in the future, we plan to develop a method to input these extra characters automatically.

IV. Experiment

To evaluate the feasibility of OSPAK, we invited 95 volunteers to use OSPAK to input a sequence of strings (passwords). We observed their input results to check whether OSPAK influences the normal password input operation of a standard user, while protecting a host against keyloggers. Because frequent logins or logouts and continuous login errors are frequently deemed as attacks by many servers, in our experiments we did not ask these 95 subjects to really log in to a server. Instead, based on the OSPAK, we created a form for the 95 experimental subjects to input a sequence of stings (passwords) provided by us. We used the form to simulate a real login form. The round-trip time between the left end and the right end of the moving foreground object is eight seconds.

Each experiment consists of two phases. In the first phase, after explaining the OSPAK method to the subjects, without making any practice, the 95 subjects immediately began their experiments. In each experiment, the system showed a string (password) above the form to ask a user to input it. For each user, 10 passwords were used in this phase. If a user correctly input a password, the system asked the user to input the next password and the input operation was deemed as a *correct input*. If a user erroneously input a password, the system asked the user to input the password again and, the input operation was deemed as a *failed input*. If a user cannot input a password correctly in 10 attempts, the system skips this password and continues with the next one. In the second phase, before doing their experiments, the 95 subjects practiced to input the 10 pre-defined strings once, and then these subjects repeated the experiments that they had performed in the first phase. The purpose of these two-phase experiments is to confirm the following two questions: (1) Is OSPAK easy to use? (2) For a user, how much improvement can be made through practice in using OSPAK? Based on the data we collected from the above experiments, we extract the information discussed in the following subsection to evaluate the feasibility of OSPAK.

A. Input Correctness Rate

We define a metric *Input Correctness Rate* (ICR) to evaluate the correctness rate of experiments that all 95 subjects performed when they input the sequence of strings (passwords) provided by us. The definition of ICR is as follows:

$$\text{ICR} = \frac{\alpha}{\alpha+\beta} \tag{1}$$

α = total number of correct inputs made by all 95 subjects. β= total number of failed inputs made by all 95 subjects.

B. Average Number of Attempts

Besides the overall correctness rate mentioned, we also calculate the average number of input attempts to correctly input a password for all 95 subjects before and after training. Fig. 1 shows the results, where σ represents the standard deviation. The average numbers of input attempts are low in both cases. After training, on average, a user only needs to input a password 1.09 times. As a result, OSPAK only increases a small number of the times a user needs to correctly input a password.

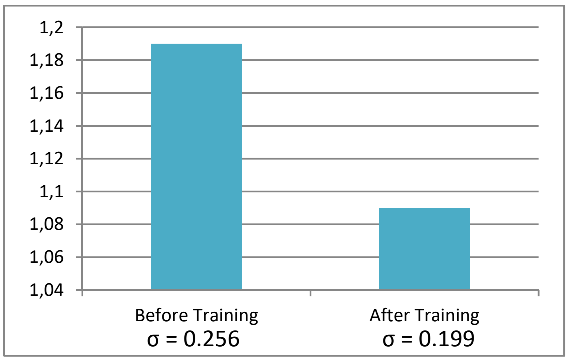

Fig. 1 Average number of input attempts to correctly input a password for all 95 subjects before and after training

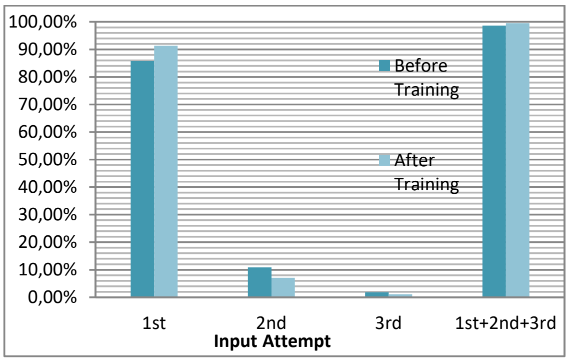

Fig. 2 The percentage of successful login attempts

To protect a system against brute-force attacks, many web servers limit the number of continuous failed login attempts that a user can make during a period of time. Continuous login failures are usually deemed as an attack and the related account will be disabled temporarily. The upper bound of continuous login failures is usually two. In other words, if a user continuously fails to log in to an account three times, the user's account will be disabled temporarily. However, on average, OSPAK consumes 0.09 more of a user's time to correctly input a password. To evaluate whether OSPAK influences the normal logins of users, we analyzed the percentage of passwords that were correctly input by the 95 subjects on their first input attempts. Similarly, we also analyzed the percentage of passwords that were correctly input by the subjects on their second input attempts and what percentage of passwords was correctly input by the subjects on their third input attempt. Finally, we calculated the percentage of passwords that were correctly input by the subjects using less than or equal to three input attempts. The results are shown in Fig. 2. After a short training period, about 91% of users can successfully log in to a system on their first input attempt, while 99% of users can correctly input their passwords within three input attempts. The analyses show that OSPAK has little influence upon normal users' logins.

According to the definition, a high input correct rate means that subjects can use OSPAK easily to correctly input passwords. On the contrary, a low ICR means that OSPAK is not easy to use for subjects. Before training, the ICR of all 95 subjects was 79%. After training, the ICR of the 95 subjects increased by 14.73%. The improvement only came from the practice of letting each subject input 10 strings. Therefore, the OSPAK method is easy to use. And users can become familiar with OSPAK quickly.

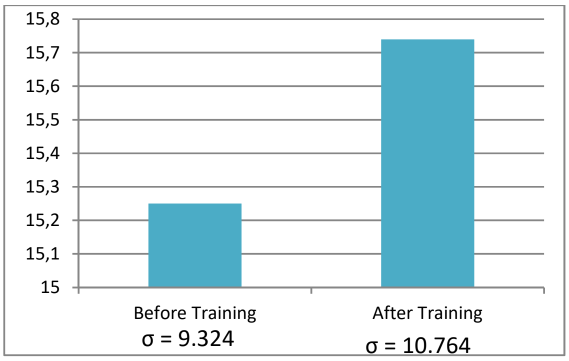

Fig. 3 Average numbers of meaningless characters inserted into passwords by new OSPAK users

C. Average Numbers of Meaningless Input

Users can use OSPAK to insert meaningless characters into their passwords to confuse keyloggers. Therefore, we analyze the average number of meaningless characters that new OSPAK users inserted into their passwords. Figs. 3 and 4 show the results. In Fig. 3, σ represents the standard deviation. The lengths of our pre-defined passwords are between seven characters to 10 characters. As shown in Fig. 3, before and after training, the numbers of meaningless characters inserted by new OSPAK users are about 15. After training, users input more meaningless characters. Hence, OSPAK indeed increases the difficulty for keylogger users to crack passwords. Fig. 4 shows the distribution of the average numbers of meaningless characters inserted into passwords by new OSPAK users before and after training. About 28% of users inserted an extra 11 to 15 characters, and about 20% of users inserted more than 31 characters in the experiments. Generally, the more meaningless characters a user inserts, the more difficulty a password cracker encounters. Hence, in the future, we plan to reminder OSPAK users to insert as many meaningless characters as possible when a moving foreground object is over a background area. Alternatively, a warning message will appear to an OSPAK user, when they do not input enough meaningless characters.

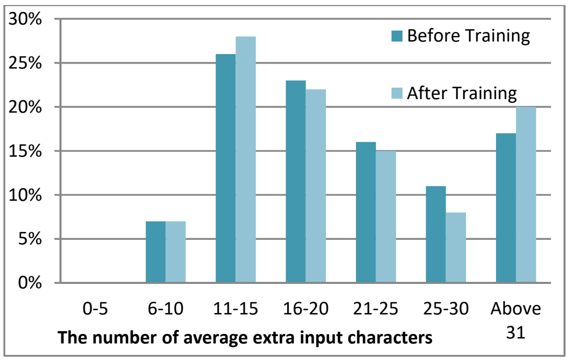

Fig. 4 The distribution of average extra input characters

V. Related Work

We propose OSPAK as a solution to keyloggers and recommend clients to use this implementation as a browser extension, which can be integrated in a web page. Nevertheless, there are several state-of-the-art solutions for protecting authentication. The latest authentication standardization initiative, FIDO Alliance, proposes an interoperable combination of asymmetric cryptography with biometrics or two-factor authentication [5]. Smartcard framework also proposes a method consisting of a proposed structure consisting of a smartcard-reader, enforcement of access control on the smart card, and protected communication [6]. A research on operation systems also talks about secure user input [7]. We may see various authentication mitigations. However, when it comes to simplicity, OSPAK is easier and more user friendly for general uses, without complicated setup or operation steps.

VI. Conclusion

As a browser extension, OSPAK can protect computers against recoding sensitive inputs, which obfuscates keyloggers with letters inserted among users' keystrokes. Moreover, we have tested seven different user-space and kernel-space keyloggers, and none of them could obtain the real passwords. OSPAK is compatible with all websites without the need of their support. To conclude, users can be protected from keystroke logging with the OSPAK method utilizing animation to visualize valid and invalid time intervals. We have tested it and the results show that OSPAK is a good choice to protect users' privacy. OSPAK can surpass nearly all keyloggers presuming the attacker will not record the screen and analyze them manually.

References

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

[2] BSI WARNING, "Questions and answers about identity theft," https://www.golem.de/news/bsi-warnung-fragen-und-antworten-zum-identitaetsdiebstahl-1401-104118.html

[3] C.-W. Hung, F.-H. Hsu, S.-J. Chen, C.K. Tso, Y.-L. Hwang, P.-C. Lin, and L.-P. Hsu, A QTE-based Solution to Keylogger Attacks, The Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2012), Rome, Italy, August 19 - 24, 2012.

[4] Jelsoft Enterprises Ltd, "What password-length do you use on most websites?" https://www.wilderssecurity.com/threads/what-password-length-do-you-use-on-most-websites.178319/

[5] FIDO Alliance - Open Authentication Standards More Secure than Passwords, https://fidoalliance.org/

[6] Bundesamt fuer Sicherheit in der informationstechnik. Technical Guideline TR-03112-1 eCard-API-Framework - Overview. Version 1.1.5 draft, 7. April 2015

[7] B. Pfitzmann, J. Riordan, Christian Stüble, M. Waidner, A. Weber: The PERSEUS System Architecture; IBM Technical Report RZ 3335 (\#93381), IBM Research Division, Zurich Laboratory, 2001