

Hardware Approach to Solving Password Exposure Problem through Keyboard Sniff

Kyungroul Lee, Kwangjin Bae, and Kangbin Yim

Abstract—This paper introduces a hardware solution to password exposure problem caused by direct accesses to the keyboard hardware interfaces through which a possible attacker is able to grab user's password even where existing countermeasures are deployed. Several researches have proposed reasonable software based solutions to the problem for years. However, recently introduced hardware vulnerability problems have neutralized the software approaches and yet proposed any effective software solution to the vulnerability. Hardware approach in this paper is expected as the only solution to the vulnerability.

Keywords—Keyboard sniff, password exposure, hardware vulnerability, privacy problem, insider security.

I. INTRODUCTION

MOST of the online payment system requires IDs and passwords to authenticate their users for one or more steps. The Internet banking service requires a strong security with more than four steps of authentication for certificate password, account password, transfer password and OTP password. The passwords in this case are all normally strings that are gathered from the computer keyboard.

Even though a strong authentication using multiple passwords is deployed, the problem has been arisen that the information from the computer keyboard can be easily stolen out of the authentication software by other malicious code.

Common objective of the keyboard security technologies is basically to gather scan codes from keyboard prior to attackers and keep them unexposed. However, there have been known opportunities for attackers to intervene gaps found on weak interfaces of software or hardware in the keyboard security framework.[3] One of the most severe problems is that there has been no reasonable solution to direct access to the keyboard interface hardware.[5] This is because that the keyboard hardware is not dispensable and open to any software running with a privileged level of access, both for PS/2 and USB interfaces.[1][2]

Severe vulnerability of the PS/2 keyboard introduced in a

Kyungroul Lee is with the Soonchunhyang University, Department of Information Security Engineering, 646 Eupnae, Shinchang, Asan, Korea (e-mail: carpedm@sch.ac.kr).

Kwangjin Bae is with the Soonchunhyang University, Department of Information Security Engineering, 646 Eupnae, Shinchang, Asan, Korea (e-mail: kjbae@sch.ac.kr).

Kangbin Yim is with the Soonchunhyang University, Department of Information Security Engineering, 646 Eupnae, Shinchang, Asan, Korea (corresponding author to provide e-mail: yim@sch.ac.kr).

prior research is related to the indispensability of the keyboard buffer in spite that it lacks atomicity and an internal status of the keyboard controller is exposed to the host software. The USB problem for keyboards is that the USB transaction buffers are memory mapped and available to any software during a frame time.

The rest of this paper is organized as follows. Section II explains the trials and errors related to the keyboard security. Section III introduces a hardware approach to solve the keyboard sniffing problem and Section IV shows a sample implementation for the hardware approach. Lastly, Section V concludes with future considerations for a practical application of the approach.

II. KEYBOARD SECURITY CHALLENGES AND FAILURES

Keyboard interface hardware is divided into two classes. One is PS/2 and the other is USB even the later is not only for keyboards. Both have serialized protocols but basics of the interfaces are different in the point that the PS/2 has an I/O based interface and the USB has a memory mapped interface.

The PS/2 Keyboard interface hardware is mainly comprised of a keyboard processor and a keyboard controller. The keyboard processor is placed inside the keyboard itself and connected to the keyboard controller, generates scan codes from user key inputs and responses to the keyboard command codes. The keyboard controller is located as a periphery device in the computer, translate scan codes to key codes and responses to the keyboard control codes.[2] The keyboard controller has a pair of buffers in both direction of input and output. The output buffer is used to transfer the scan code if a key is pressed, which is readable for any software and causes keyboard sniffing problem.

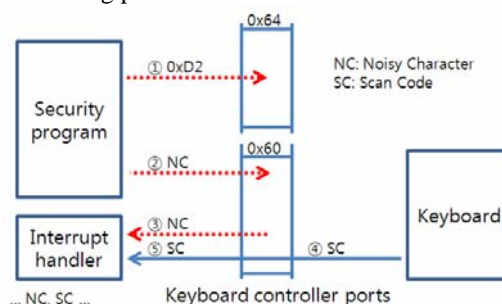


Fig. 1 Security software can deceive a sniffer by generating noisy characters that are mingled into the password characters from keyboard. However, this behavior is easily understood by an attacker through monitoring a flag named C/D bit in the keyboard controller

Several researches had tried to find solutions to the direct sniffing of the keyboard information. Most reasonable one of them deceives sniffing software by generating noisy data and mingling them into real keyboard information, as shown in figure 1. However, a research recently introduced a vulnerability of the keyboard controller. According to the research, a flag named C/D bit in the status register on the keyboard controller enables an attacker to steal passwords from the keyboard even the mingling algorithm is applied.[1][2]

Major difference of the USB from the PS/2 for keyboard interface is that it has a USB host controller and a root hub instead of the keyboard controller. The USB host controller is accessed through a set of registers and a memory mapped structures and buffers as shown in Fig. 2, as though some trivial differences exist between versions.

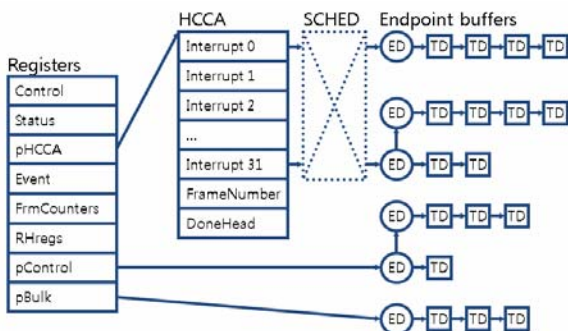


Fig. 2 USB devices are interfaced to the computer through a host controller with a set of structures that are readable and writable. The typematic information from the USB keyboard is easily sniffed by an attacker through monitoring the transaction buffers

The pHCCA is a readable and writable register containing the address of the host controller communication area that is mapped into the primary memory of the computer, in which a set of pointers to the linked lists for transaction buffers resides. Because the pHCCA register, the HCCA as well as the transaction buffers are all readable and writable for any software, the keyboard information through the USB keyboard is easily sniffed and stolen.[3]

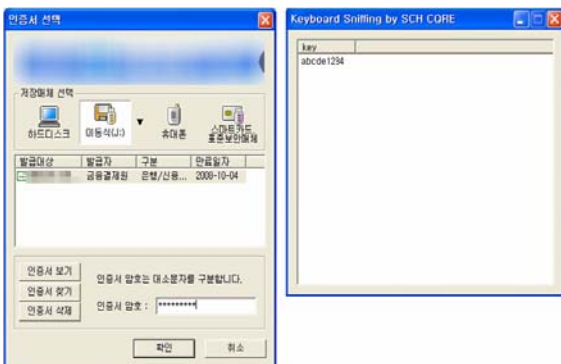


Fig. 3 Result screen. The left one is a public interface of the authentication module for the certificate used in the Internet banking services. The right one is the client screen of the keyboard sniffing agent (device driver). The password inputted from the keyboard is acquired easily through the hardware vulnerability on both PS/2 and USB interface

III. KEYBOARD SECURITY IN HARDWARE

Security technologies are primarily involved in the field of cryptography. Because of the reason, early researches on the secure keyboard were engaged in the complex crypto issues about keyboard information.[5][6] Even though a chick cryptic technologies on the keyboard, they are not effectively applied to real input equipment because the complexity of the technologies is beyond the limit of the keyboard implementation in the points of economic feasibility or practical obstacles.

When data are transferred from one party to another, encrypted secure channel is required when a public channel is used for communication because the data could be acquired by others without permission. In the case that a secrete way could be found between allowed parties, there wouldn't be much necessary to require complex cryptography.

Hardware engineers have not been much considering how the interface in their design would be immune to tampering by software through possible vulnerability from its own; instead they focus on tamperproof of the crypto algorithms only or consider resource economy or access readiness. Most important thing for the security hardware engineers to consider in their design is to find the secrete channel for communication to lessen the complexity of the crypto algorithm used.

PS/2 Keyboard interface looks not designed considering security mentioned earlier. However, resource economy done on the PS/2 keyboard interface fortunately leads to possibility to find a easy way to distribute a secrete key and random vectors for keyboard data encryption.

Writing data into port 0x60 of the keyboard controller is a hidden and disposable action for dedicated software because the port is organized as a one way volatile interface in hardware. Because of the reason, the data written is bound only to the keyboard itself and never found in any other place and accessed except the dedicated software.

The number of possible keyboard scan codes is double of the number of possible key caps on the keyboard, which leads to not more than 256. This means that the scan code for one keycap is required to be changed in time randomly. One time key scheme is a good candidate for this requirement because it is easy for dedicated software and the keyboard to share the keys through the port 0x60.

IV. IMPLEMENTATION EXAMPLE

To utilize the method explained above, the keyboard should be re-programmed. It could be fine to incorporate security unction into keyboard itself. However, it is usually not acceptable because some people seldom want to replace their accustomed keyboard. A reasonable approach is to place a security hardware module between the host and the keyboard.

The security hardware module deals with the keyboard like as a normal host and implements a security function with the host security software. Figure 4 shows the internal function blocks implemented in the security hardware module including the connectors both to the host and the keyboard. Figure 5

shows the assembled security hardware module. The connector side of the module has a facility to reprogram the security function, which could be used when the security algorithm is exposed on the host side.

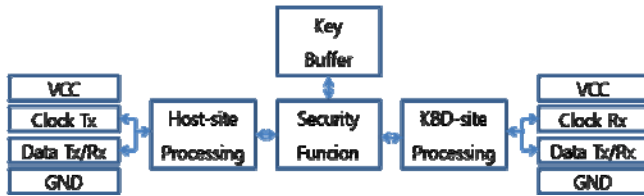


Fig. 4 Internal function blocks of the security hardware module. It mainly consists of three different functions; the host-site processing, the keyboard-site processing and the security function. Security function needs just a simple randomized shared key cryptography because of the disposability of the PS/2 keyboard interface



Fig. 5 Implemented and reprogrammed security hardware for PS/2 keyboard as an example. USB keyboard is not considerable for a secure keyboard because the USB host controller has no disposable interface between the host and the target device and it is adapted to the PS/2 interface through a combo converter

V. CONCLUSION

This paper introduced a hardware based solution to the password sniff problem caused by the severe intrinsic hardware vulnerabilities of the keyboard hardware. Because the vulnerabilities still have no software based solution and even any possible candidates, it is required to have some types of reasonable hardware solutions. Fortunately the PS/2 keyboard interface on the host side has a disposable interface, only a simple shared key crypto algorithm is enough though it needs to have a one-time randomized key chain. However, because the USB interface is getting common for the keyboard, which is too weak for security applications, it is required to keep or deploy a disposable interface such as PS/2 in the PC platform or various new trials not using keyboard strings for passwords.[4]

REFERENCES

- [1] Taeyoung Jung, Kangbin Yim, "Countermeasures to the Vulnerability of the Keyboard Hardware," *Journal of the Korea Information Security and Cryptology*, Vol. 18, No. 4, pp.187-194, Aug., 2008.
- [2] Kwangjin Bae, Kangbin Yim, "Analysis of an Intrinsic Vulnerability on Keyboard Security," *Journal of the Korea Information Security and Cryptology*, Vol. 18, No. 3, pp.89-95, Jun., 2008.
- [3] Kangbin Yim, "A fix to the HCI specification to evade ID and password exposure by USB sniff," *Proceedings of APIC-IST 2008*, pp.191-194, 2008.12.18-19.
- [4] Taeyoung Jung, Kangbin Yim, "A new image-based login method against keyboard sniff," *Review of the Korea Information Security and Cryptology*, Jun., 2008.

- [5] Kangbin Yim, "Keyboard Security," *Workshop on Ubiquitous Information Security*, May, 2008.
- [6] Linda D. Paulson, Key "Snooping Technology Causes Controversy," *IEEE Computer*, pp.27, Mar. 2002.