Unprivileged Black box Detection of User Space Keystroke Harvesting Malware

J.RosliJeba[1], PG Scholar,

T.C.Subbulakshmi[2], Assistant Professor,

Department of Information Technology
Francis Xavier Engineering College, Anna University.
Vannarpettai, Tirunelveli ,Tamilnadu,India.

Abstract— Key loggers are used on a machine to monitor the user activity by logging keystrokes and delivering them to a third party. The main goal is to prevent user-space key loggers from stealing confidential data originally intended for a legitimate foreground application. Therefore a new detection technique has been proposed that simulates carefully crafted keystroke sequences in input and observes the behaviour of the key logger in output to identify it among all the running processes. The proposed detection technique is implemented in C#, it runs as an unprivileged application for the Windows OS. An unprivileged black box approach for accurate detection of user space key loggers has been devised by correlating the input (keystrokes) with the output (I/O patterns produced by the key logger).

Keywords— Invasive software, keylogger, security, black-box

I. INTRODUCTION

Key stroke harvesting malware are implanted on a machine to intentionally monitor the user activity by logging keystrokes and eventually delivering them to a third party [1].

A fully unprivileged key logger can be implemented by a simple user space process. It can be used for both legal and illegal purposes. It is perfectly legal so long as it is installed on the owner of the computer. In the workplace, key loggers are often used to monitor staff when they are using company computers. It is also useful in the home, where they may be used to monitor the activities of children or teenagers online. Keeping children safe on the Internet is extremely important [3]. It is illegal if this is installed on a computer that does not belong to owner. In such illegal cases, the detection of whether the keystroke harvesting malware is present in a system or not is mandatory.

Hypervisor-based key loggers (e.g.BluePill) are the straightforward software evolution of hardware-based keyloggers, literally performing a man-in-the-middle attack between the hardware and the operating system (OS). Kernelkeyloggers come second in the chain and are often implemented as part of more complex rootkits. In contrast to hypervisor-based approaches, hooks are directly used to intercept buffer processing events or other kernel messages [1]. It is important to notice that a user-space key logger can easily rely on documented sets of unprivileged APIs commonly available on modern operating systems (OSs).

This is not the case for a key logger implemented as a kernel module. In kernel space, the programmer must rely on kernel-level facilities to intercept all the messages dispatched by the keyboard driver, undoubtedly requiring a considerable effort and knowledge for an effective and bug-free implementation.

Furthermore, a key logger implemented as a user-space process is much easier to deploy. since no special permission is required. Despite the rapid growth of key logger based frauds (i.e., identity theft, password leakage,

etc.), not many effective and efficient solutions have been proposed to address this problem.

In order to detect the malware running as unprivileged user-space processes a new approach is proposed. To match the same deployment model, the approach is entirely implemented in an unprivileged process. It is completely black-box, i.e., based on behavioural characteristics (system calls or library calls invoking) common to all keyloggers.

It does not rely on the internal structure of the keylogger or the particular set of APIs used. Therefore, the solution is of general applicability. It simulates carefully the crafted keystroke sequences in input and observes the behavior of the keylogger in output to unambiguously identify it among all the running processes. Thus if such malware is detected, user using the system will be in a safer side without being their confidential information getting captured and used by an attacker.

A. Paper Organization

The rest of the paper is organized as follows. In Section II the system architecture is shown. Section III describes the proposed approach. Section IV, V and VI describe the modules, results and finally concludesthe paper.

II. SYSTEM ARCHITECTURE

The overall system architecture is given in fig 1.The architecture of the Detecting Keystroke Harvesting Malware involves various steps in order to attain the goals. The steps include:

A. Pattern Translator:

The role of the pattern translator is to transform an AKP into a stream and vice-versa, given a set of configuration parameters [1]. A pattern in the AKP form can be modelled as a sequence of samples originated from a stream sampled with a uniform time interval. This assumption does not always hold in practice and the detection algorithm has to consider any possible scale transformation between the input and the output pattern.

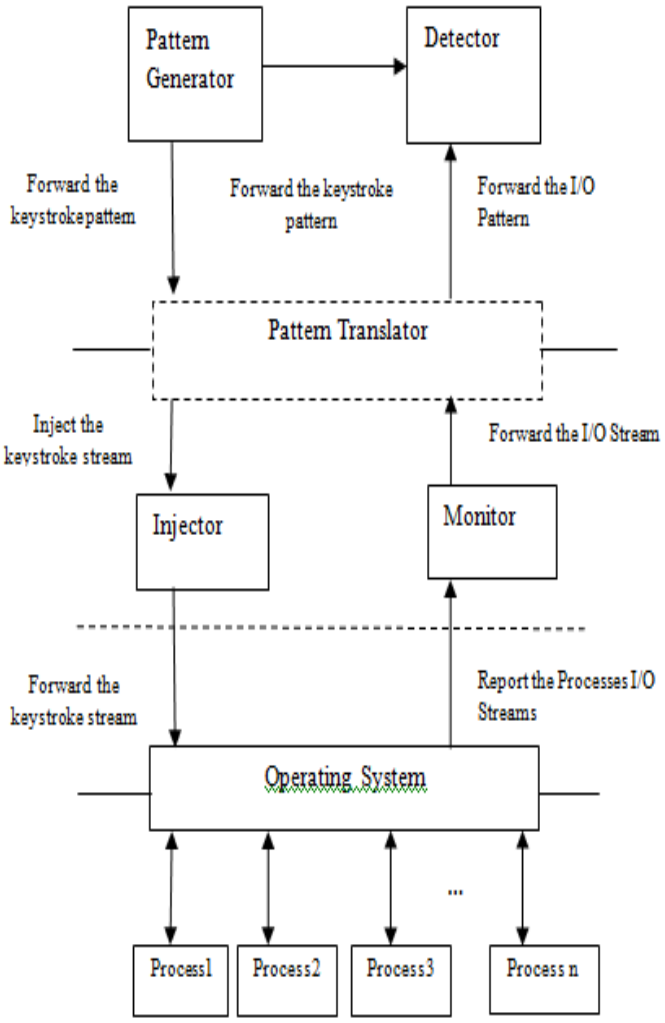

Fig .1.System Architecture

B. Injector

This injects the captured key into a text file at any particular location of the user's computer. It is also called as logging captured key.

C. Monitor

The monitor is responsible to record the output stream of all the running processes. As done for the injector, it allows only unprivileged API calls. In addition, it favours strategies to perform real time monitoring with minimal overhead and the best level of resolution possible. Finally, these projects are interested in application-level statistics of I/O activities, to avoid dealing with file system-level caching or other potential nuisances. Fortunately, most modern operating systems

provide unprivileged API calls to access performance counters on a per-process basis [4].

To construct the output stream of a given process, the monitor queries this piece of information at regular time intervals, and records the number of bytes written since the last query every time. The proposed technique is obviously tailored to Windows-based operating systems. None the less, this project point out those similar strategies can be realized in other OSes; both Linux and OSX, in fact, support analogous performance counters which can be accessed in an unprivileged manner; the reader may refer to the io top utility.

D. Detector

Detector is the assignment of a label to a given input value. An example of pattern recognition is classification [14], which attempts to assign each input value to one of a given set of classes (for example, determine whether a given email is "spam" or "non-spam"). However, pattern recognition is a more general problem that encompasses other types of output as well.

III. PROPOSED APPROACH

This paper proposes a new approach to detect key loggers running as unprivileged user-space processes. To match the same deployment model, this technique is entirely implemented in an unprivileged process. As a result, this solution is portable, un intrusive, easy to install, and yet very effective. In addition, the proposed detection technique is completely black-box, i.e., based on behavioural characteristics common to all keyloggers [1]. In other words, this technique does not rely on the internal structure of the keylogger or the particular set of APIs used. For this reason, this solution is of general applicability. This paper prototyped the approach and evaluated it against the most common free keyloggers. It is also evaluated the impact of false positives in practical scenarios. In the final part of this paper, it will validate the approach with a home grown keylogger that attempts to thwart detection technique. This project must carefully deal with possible data transformations that may introduce quantitative differences between the input and the output patterns.

IV. MODULES

For an efficient detection technique which prevents user from entering their confidential information to the attacker without their knowledge, the process has been divided into separate modules. They are as follows:

A. Attacker Login Module

This is the module where an attacker enters their password to login to run their Program that captures all the keys without user's knowledge as they types in the keyboard. Login is needed for an attacker to maintain privacy of his or her code. Privacy is needed due to the following reasons. Attacker is in need of performing physical attack first of all to the system used by the user, the same program should not be taken by any other third party and used by them to harvest user's confidential information. The keystroke logger program records each keystroke the user types and uploads the information over the Internet periodically to whoever installed the logger program.

B. Monitor Module

As the name indicates, monitor module observes and supervises activities in progress. Once when the attacker login and run a program that harvest confidential information in a user's computer by performing physical attack, the program starts running and captures all the keys pressed by a user. This is also called as key capturing.

The system provides language-independent keyboard support by using the language-specific keyboard layout currently selected by the user or the application. The keyboard device driver receives scan codes from the keyboard, which are sent to the keyboard layout where they are translated into messages and posted to the appropriate windows in your application.

Assigned to each key on a keyboard is a unique value called a scan code, a device-dependent identifier for the key on the keyboard. A keyboard generates two scan codes when the user types a key—one when the user

presses the key and another when the user releases the key.

After translating a scan code, the keyboard layout creates a message that includes the scan code, the virtual-key code, and other information about the keystroke, and then places the message in the system message queue. The system removes the message from the system message queue and posts it to the message queue of the appropriate thread. Eventually, the thread's message loop removes the message. It is passed to the appropriate window procedure for processing.

C. Injector Module

This module injects the captured key into a text file at any particular location of the user's computer. It is also called as logging captured key. Log files will be placed either in a Predefined (default) folder or in a customized folder. The text file is appended every time when the keys are about to store. If it is not appended, the storage of captured keys will be replaced every time. This file may be kept hidden to user by an attacker which prevents user from detecting an attacker. Thus it act as a record for all the data's including confidential information of user. Attacker will make use of this record in order to harvest user's confidential information.

The folder of the log files will be saved on disk and will be hidden from the computer user (password protected/invisible). Furthermore, access to the folder during system activity (monitoring in progress) will be forbidden.Not to cause a conflict with the system. Only one logs-folder will be at any time on the computer. Description of the log files includes its size and type log file. The type of the log file may be a text file etc.

D. Key Transfer Module

As the keys getting captured and stored in a text file (record), it may be transferred to an attacker's mail id in the mean while for every particular interval of time. A timer is set for time interval. Log delivery via email has been done by using SMTP. Keylogger requires an active Internet connection to report your sensitive personal data back to a third party. A keylogger program typically sends packets of personal information through an unsecured port in your Internet connection to a database or remote location, where criminals can monitor all activity or set up searches for sensitive financial data or other personal information.

E. Detection Module

Detecting the presence of hardware keylogger is quite easy as it requires only the physical examination of the computer what we are using. In case of detecting the keystroke harvesting malware which comes under software keylogger, various simple methods are there. It includes:

A user has to take help of a simple process to detect it and to eradicate it from a system as it remains installed in more than hundred places of a computer. However, to identify whether a keylogger is running or not, while using the desktop, right click the task bar and then go to the task manager. Alternately, cntrl + Alt + Del can be pressed concurrently to undo the task manager. The task manager will display a list of the applications currently running on the machine. Then go to the process tab and it will displays the information on both visible and invisible program.

Use 3rd party software. Since keyloggers are meant to NOT be detected easily, it may be necessary to use a keylogger detection program. Such an application will go over your system and verify the presence a keylogger and provide details about its name, where it's located on your system and so on.

Although those detection methods are available, in case of first method, it is difficult for user who is unaware about keylogger. Coming to second method, antivirus which detects such presence may be expensive and finally for the third method, special installation is needed for detection.

In order to overcome all those difficulties, a detection technique that is completely based on the behaviour of the keylogger has been proposed. If the key stroke harvesting malware is present in a particular system, once when the user started pressing his first key, this technique gives the warning symbol to the user stating

that "your keys are getting captured". Also it shows the model of how the keys are getting captured by an attacker. If no such malware is present, when the user started pressing the key, a notification that your machine is safe to use will be given to user.

V. EXPERIMENTAL RESULTS

In this approach a new detection technique is used.The detection technique can be used to detect any kind of keyloggers such as keylogger monitoring, email keylogger and so on. It is detected based on the behaviour of the key stroke harvesting malware.This technique has been successfully evaluated prototype system against the most common free keyloggers.

TABLE I

KEYLOGGER	DETECTION
RefogKeylogger Free 5.4.1	✓
Best Free Keylogger 1.1	✓
Iwantsoft Free Keylogger 3.0	✓
Actual Keylogger	✓
RevealerKeylogger Free 1.4	✓
Virtuoza Free Keylogger 2.0	✓
Quick Keylogger 3.0.031	✓
Tesline Kid Logger 4.1	✓

VI. CONCLUSION

An unprivileged black-box approach for accurate detection of the most common key loggers i.e., user space keyloggers. The behavior of a keylogger by surgically correlating the input (i.e., the keystrokes) with the output (i.e., the I/O patterns produced by the keylogger). The detection technique can be used to detect any kind of keyloggers such as keylogger monitoring, email keylogger and so on. It is detected based on the behaviour of the key stroke harvesting malware. Sub sequent, an implementation of detection technique on Windows, arguably the most vulnerable OS to the threat of keyloggers. This technique has been successfully evaluated prototype system against the most common free keyloggers. This approach considerably raises the bar for protecting the user against the threat of keyloggers.

ACKNOWLEDGMENT

The support of Mrs T.C.Subbulakshmi2, AssistantProfessor is gratefully acknowledged.

REFERENCES

[1] S. Ortolani and B. Crispo, "Noisykey: Tolerating KeyloggersviaKeystrokes Hiding," Proc. Seventh USENIX Workshop Hot Topics in Security, 2012.

[2] San Jose Mercury News, "Kinkois Spyware Case Highlights Risk of Public Internet Terminals," 2012.

[3] N. Grebennikov, "Keyloggers: How They Work and How to Detect Them," 2012.

[4] M. Vuagnoux and S. Pasini, "Compromising Electromagnetic Emanations of Wired and Wireless Keyboards," Proc. 18thUSENIX Security Symp., pp. 1-16, 2009.

[5] T. Holz, M. Engelberth, and F. Freiling, "Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones," Proc. 14th European Symp. Research in Computer Security, pp. 1-18, 2009.

[6] D. Brumley, C. Hartwig, Z. Liang, J. Newsome, D. Song, and H. Yin, "Automatically Identifying Trigger-Based Behavior in Malware," Advances in Information Security, vol. 36, pp. 65-88, 2008.

[7] J. Han, J. Kwon, and H. Lee, "Honeyid: Unveiling Hidden Spywares by Generating Bogus Events," Proc. IFIP 23rd Int'l Information Security Conf., pp. 669-673, 2008.

[8] Y. Al-Hammadi and U. Aickelin, "Detecting Bots Based on Keylogging Activities," Proc. Third Int'l Conf. Availability, Reliability and Security, pp. 896-902, 2008.

[9] A. Moser, C. Kruegel, and E. Kirda, "Exploring Multiple Execution Paths for Malware Analysis," Proc. IEEE 28th Symp. Security and Privacy, pp. 231-245, May 2007.

[10] E. Kirda, C. Kruegel, G. Banks, G. Vigna, and R. Kemmerer, "Behavior-Based Spyware Detection," Proc. 15th USENIX Security Symp., pp. 273-288, 2006.

[11] L. Goodwin and N. Leech, "Understanding Correlation: Factors that Affect the Size of r," Experimental Education, vol. 74, no. 3,pp. 249-266, 2006.

[12] L. Zhuang, F. Zhou, and J.D. Tygar, "Keyboard Acoustic Emanations Revisited," ACM Trans. Information and System Security, vol. 13, no. 1, pp. 1-26, 2009.

[13] M. Aslam, R. Idrees, M. Baig, and M. Arshad, "Anti-Hook Shield against the Software Key Loggers," Proc. Nat'l Conf. Emerging Technologies, pp. 189-191, 2004.

[14] G. Kochenberger, F. Glover, and B. Alidaee, "An Effective Approach for Solving the Binary Assignment Problem with Side Constraints," Information Technology and Decision Making, vol. 1,pp. 121-129, May 2002.

[15] J.L. Rodgers and W.A. Nicewander, "Thirteen Ways to Look at the Correlation Coefficient," The Am. Statistician, vol. 42, no. 1, pp. 59-66, Feb. 1988.