A Robust Technique of Anti Key-Logging using Key-Logging Mechanism

Muzammil M. Baig[1], W. Mahmood[2]

[1,2]Muzammi M. Baig, W. Mahmood, Al-Khawarzmi Institute of Computer Science, University of Engineering and Technology, Lahore, Pakistan[Ψ].
e-mail: (muzammil,wmahmood)@kics.edu.pk

***Abstract*—System security and privacy always have to face new confronts. Continuous updates in the operating systems and anti-virus applications strive to amplify the system security level. In recent years 'Key-Loggers' have proved to be one of the prevalent intimidations to security and privacy. Key-logger is a surreptitious surveillance application, which is used to keep record of user's activities on the computer in various ways like keyboard logging, screen logging, mouse logging and voice logging, completely in imperceptible mode. Although key-loggers can also be used for prolific purposes but due to the tremendous increase in the Internet usage, the caustic use of key-loggers simply surmounts its advantages. Key-loggers have gained so much supremacy in their execution that they have become a serious intimidation to the privacy and security of a computer. The fact which makes the key-loggers more perilous is their undetectable nature against anti-virus and spy-where applications. This paper discusses some existing techniques of fortification against key-loggers and also exemplifies a new technique along with its proved advantages.**

***Index Terms*— anti key-logging, non-signature based scanning, system hooks, application hooks, software key-logger**

I. INTRODUCTION

The Race between 'Privacy and Piracy' has always been a source of impetus for both researchers and hackers. In this never-ending combat, the latest challenge is the key-loggers, both software [15] and hardware [16]. Software based key logging is a familiar constituent of *Trojan horses*, that are often installed by gaining physical access to the computer or by downloaded programs. Hardware key-loggers are in common practice for publicly shared systems and they are attached or placed inside the keyboard. Although our proposed new technique of anti key-logging pledges protection against both the software and hardware key-loggers but in subsequent sections expression key-logger refers to software key-loggers, otherwise if specified. Key-loggers have small footprint in terms of memory and processor utilization and this property makes them practically untraceable for the user. Most of the key-loggers initiate their process execution using the name of any system service routine. In this way user can't distinguish between key-logger processes and any other system routine [4]. Most peril property of the key-loggers is their barely imperceptible nature against anti-virus applications.

In subsequent section the internal architectural details of the key-logging applications are elucidated to find out the reason of their escape from anti-virus applications. Section 2 and 3 depicts an overview of background and related work, including the review of *'signature based'* and *'non-signature based'* anti key-logging techniques. Section 4 illustrates our new technique of the anti key-logging and most importantly some of the test results are also taken into account in section 5. Future work and conclusion are summed up in section 6.

II. KEY-LOGGING ARCHITECTURE

It is very easy to get a key-logger, as most of them are freely available for download from Internet [5, 6]. Even one can find list of many from any search engine like Google. This easy access of key-loggers creates more perils toward personal privacy, as most of key-loggers also provide some means of remote installation in completely hush mode. Most of the free software, available on Internet, also have hidden spy-where along them [14]. Once a key-logger gets installed, then one can imagine that all of the typed keys become unsecured. It is not necessary for the intruder to have physical access of a system both for installation and key-log viewing [12]. Some prevailing key-loggers [13] can perform these tasks remotely and one user can never came to know that all of its keys are not only logged but also sent to intruder through e-mail.

It is a fact that the development of anti-virus applications has enormously increased. Regular updates of anti-virus application are also considered a reliable source for system security [9]. Even then most of the key-loggers get the room for execution in the system. This is due to the fact that essentially key-loggers are not viruses [7]. Next section illustrates this fact in more detail.

Deep understanding with operating system [11] architecture divulges that key-loggers do not employ with any such type of code which can be considered as malevolent one. First the architectural details of the key-loggers are explained then the reason is identified which puts them aside from anti-virus applications.

[Ψ] Part of this research work was funded by Techno-soft Inc. USA
www.techno-soft.com

A. Key-Loggers and System Hooks

The development of a simple key-logger is an easy task with respect to system programming. Infect intruders have to put more effort on its stealth execution functionality. Key-logging just necessitates low level knowledge of system hooks [8]. In any system, whenever a keyboard key is pressed a specific hardware interrupt is sent to the running system. This interrupt is received by the *'system level message queue'* along with the specific *'key value'* [8, 11]. System tracks the focused application at the time when the keyboard interrupt was generated and passes the key value to the *'application level message queue'* of that specific focused application. Now it's the responsibility of that application to handle this key accordingly. If system does not find any specific focused application, it simply discards that key.

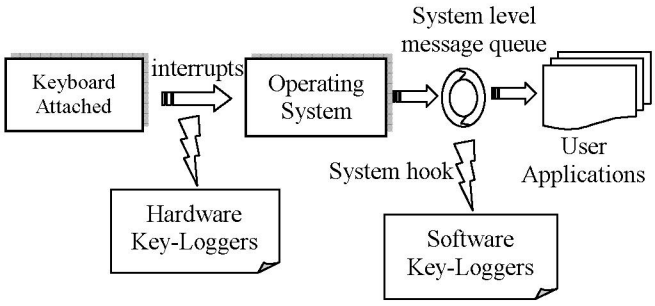

Fig. 1- Key-Loggers in system hierarchy

Key-loggers operate at system level and they hook the system level message queue. All the input interrupts are directly received by system level message queue and if any application gets the access of this system queue then it just has to apply a simple filter to get the desired keys. That's all what key-loggers do. Figure-1 illustrates the working of key-loggers in more detail.

B. Key-Loggers and Anti-Viruses

The hook of system level message queue, as key-loggers do, is not an immense task and many application programs utilize this feature too. Almost all of the modern applications hook the system level message queue during their normal course of execution. Application *'hot-keys'* is one of the best example of system message queue hook. Hot-keys refer to the application operations performed by keyboard commands like typical ctrl+s command used in most of the applications. The hook of system level message queue by any application (like hot-keys) does not guarantee that the specific application is doing key-logging. Key-loggers get the advantage of this fact and use this feature maliciously. Other applications just discard the key after performing the appropriate action but key-loggers do vice versa, they just save the keys in a log file.

In our experiments, we tested the best selling eight key-loggers with latest Norton anti-virus, updated till 09 September 2006. Some of key-loggers are so powerful that they are even invisible in system process viewer [3]. None of them was captured by anti-virus; this is due to the fact that only the system level message queue hook can't be considered malicious. If such a hook is taken into consideration then the next problem is how to distinguish between key-logging and non-key logging applications, both using the system-level hook. This scenario is described in more detail in subsequent section *'Non-Signature Based Scanning'*.

III. BACKGROUND AND RELATED WORK

This section provides a brief background for our anti key-logging technique and highlights related work. First, we review the '*Signature Based Scanning*' and '*Non-Signature Based Scanning*' mechanisms of anti key-logging.

A. Signature Based Scanning

This technique is illustrated with the reference of anti-virus applications and all the commercially available anti-viruses implement this mechanism for virus detection. In this mechanism every anti-virus application maintains its repository of virus definitions for discovered viruses. These virus definitions consist of checksums, also called signatures, for each known virus. These signatures are basically a specific sequence of malicious instructions and are used to do some ghastly task. Interested readers are referred to [1] for the computation of virus checksums. Anti-virus applications compare each file of the system against the known virus signatures. The biggest disadvantage of this technique is that it has nothing to do against those viruses, whose signatures do not exist in anti-virus signature repository [10]. Meanwhile a new virus has open opportunity to do anything with the system until the anti-virus application gets the update for that virus. Due to this fact anti-virus applications require regular updates normally daily and in worst case hourly. As described early, key-loggers do not have any specific malicious piece of code so they also don't have any specific signature. This fact provides them big whole in system security and prevents any anti-virus to detect them.

B. Non-Signature Based Scanning

This mechanism is also called '*Behavioral Scanning*'. In this technique instead of looking for the specific file signature, the behavior of the application is scrutinized, such as modification of system files and access to specific memory area. A variation of this technique is already implemented for anti key-logging [7]. But our experiments showed many problems with it. This is due to the fact that the behavior of the key-loggers is same like those applications who hook the system message queue during their normal itinerary of execution. So when this non-signature based scanning technique is applied to detect key-loggers then all those applications are also detected as malicious which hook system message queue for typical purposes not for key-logging. It is really troublesome for anyone to distinguish between key-loggers and other applications because almost all key-loggers pretend themselves as useful processes. Although such mechanisms are also suggested which tries to maintain the list of '*well-known*' processes and these processes are

ignored during non-signature based scanning. But our tests did not confer the success of this mechanism because some of key-loggers renamed themselves as system native processes, creating a possibility that any of the key-logger can be ignored during non-signature based scanning. Moreover there is always a possibility that some of the upcoming key-loggers might be so powerful that even the non-signature based scanning may fail to fabricate desired success [2].

Both of these techniques (signature and non-signature based) have nothing to do against the hardware key-loggers because such hardware key-loggers have no direct interaction with system. Such key-loggers are just attached or hidden within the keyboard and log each of the pressed key.

IV. ALGORITHM DESCRIPTION

This section describes our new mechanism of anti key-logging. First the basic model of the algorithm is described then analysis section highlights the technical details of the algorithm. In the end some of the test results are also included.

A. Basic Model and Architecture

Through understanding with the key-logging technique reveals that the focus of the key-loggers is on system message queue. This is also a fact that due to system level execution key-loggers track the key when it is received in system message queue. If somehow the keys are transferred to focused application bypassing the system message queue then we can assure the anti key-logging. This description has the main focus in our scheme.

The message passing architecture, as described in section II, divulges that all the messages are transferred from system message queue to application level message queue. Every application sustains its own message queue at application level. This application level message queue has nothing to do with the system level message queue. An application can never request to system message queue for a specific message. It's the sole responsibility of the system to manage message sending among different applications. Applications only perform appropriate actions after receiving the messages.

Some of our experiments showed that application level message queues can also be hooked by other applications. Rather it's easier at implementation level to hook the message queue of an application. In our scheme of anti key-logging we get the advantage of two main things:

- An application can't distinguish between a message send by the system message queue or by another process using application level hook.
- Application level hook does not conflict with the system message queue.

So, the simplest description of this new technique is to hook the application message queue and send those specific keyboard interrupts to application message queue which are actually send by the system message queue. In this way, we create a virtual keyboard application which bypasses the system message queue and directly posts the keyboard messages to specific application message queue using application level hook. After receiving these keyboard messages, application performs the appropriate actions accordingly as if it received the actual hardware interrupt through system message queue. Figure-2 shows the basic model of this new scheme of anti key-logging. In this figure *message 'a'* is transferred through system level message queue so it can be caught by key-loggers. Whereas the *message 'b'* is directly sent to the application message queue bypassing the system message queue. Application performs appropriate actions against both messages regardless of their source.

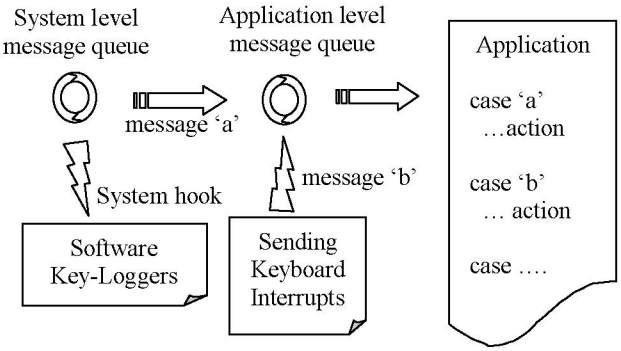

Fig.2- Anti-Key logging robust architecture

B. Anti Key-Logging Analysis

This section describes that how this new technique pledges the anti key-logging against the key-loggers. The section II described that the key-loggers protect themselves from anti-viruses. Our new technique inherits many ideas from that section, infect our technique also implement the same mechanism which key-loggers employ but our focus is on application level hook.

In this technique we purpose that we should directly hook that specific application level message queue which receives keyboard interrupts. Even for this purpose, it's not necessary for that application to have input focus too. Now after hooking the application's message queue, simply post the same keyboard interrupt messages which are sent by system level message queue. As our different experiments have already shown that application message queue can't distinguish between the source of these messages, so it simply treats them as if they are sent by system message queue and performs appropriate actions against them. Now the system level message queue is bypassed so none of the key-logger will be able to capture this input.

V. TEST RESULTS

In this section the test results of the new technique against top key-loggers are presented. We selected the best selling key-loggers for testing purposes. Table 1 shows the details of these selected key-loggers along with their version and vendor details. We developed a prototype of

this new technique in C++ under WinXP professional Service Pack 2. Using this prototype, we hooked the message queue of an application. This hooked application is capable of receiving keyboard inputs. Next our prototype application posted the interrupts of some predefined keyboard keys in the application level message queue of the hooked application. The hooked application proceeded accordingly on these keyboard interrupts as if the interrupts were sent by the system level message queue. These experiments also revealed that system processes are un-aware from the source of interrupts posted in their application level message queues. Due to this fact, when our prototype application posted the keyboard interrupts in message queue of hooked application, the hooked application accepted and responded accordingly as if they were sent by actual keyboard through system level message queue.

Table-1. Tested Key-loggers

Key-Logger Name	Company	Version
Invisible Keylogger	Cyberwire, LLC. Florida USA	1.3
PC ACME	Raytown Corporation LLC, Camden, DE USA	6.3
Ardamax Keylogger	Ardamax software, Paris, France	1.8
Perfect Keylogger	Blazing Tool Software, Berlin, Germany	1.6
Family Keylogger	Kmint Software, Zaporizhzhya, RU USA	2.8
Keyboard Spectator	RegFog Software, Pskov, Russian Federation	3.30
Win-Spy	BC Computing Team, Kuala Lumpur, Malaysia	5.6.1
Advance Keylogger	Eltima Software, Berlin, Germany	1.7

We repeated these experiments on different applications like transferring username and password to Yahoo!® Messenger (v8.0.0.701) and putting URL in the address bar of Microsoft® Internet Explorer (v6.0.2900.2180). In further experiments each of the key-loggers was executed individually during the whole procedure of data transfer and in last phase of our current experiments, all of the key-loggers were executed concomitantly during the same task of data transfer.

After all of our experiments, the log file of each key-logger was explored and none of them captured the data transfer between our prototype application and the hooked applications i.e. messenger and internet explorer. These test results also verify the strength of the algorithm, described in section IV.

If any new key-logger tries to hack our system then such a key-logger will have to hook the application level message queues of all running processes, instead of hooking single system level message queue. Hooking each or even some of important processes individually and constantly by an application level key-logger is expensive with respect to system resources both CPU and memory, making the detection easy for such application level key-logger. Moreover all other application level hooks can also be disabled before using our system to ensure that no application level key-logger gets rid of our data transfer.

VI. CONCLUSION AND FUTURE WORK

In this paper, the issue of key-loggers is addressed and a new technique of anti key-logging is proposed. This new technique not only provides the protection against software key-loggers but the appropriate working of the targeted application is also guaranteed. The architecture and analysis of this technique shows its effectiveness and test results prove its strength. These test results guarantee the anti key-logging against all commercially available key-loggers and its system message queue '*by-pass*' mechanism assures the same protection from upcoming key-loggers.

This paper emphasizes on the protection from key-loggers not on their detection. This is due to the fact that both signature based and non-signature based detection mechanisms do not pledge the full protection. Although this technique can suffers from input time complexity problem where lot of data input is required, but it is common practice that most important data (password, credit card number, PIN number) require less amount of input. Future work will focus on the implementation of this technique in other platforms especially UNIX-based systems and also decreasing the worst case time complexity for relatively large data inputs.

VII. REFERENCES

[1] Doug Varney, "Adequacy of checksum algorithms for computer virus detection", Proceedings of the 1990 ACM SIGSMALL, pp 280-282

[2] Aaron Weiss, "Spyware be gone!", *netWorker* ACM Press New York, NY, USA. March 2005, pp, 18-25.

[3] Derek Uluski, Micha Moffie, David Kaeli, "Characterizing antivirus workload execution", *Workshop on Architectural Support For Security and Anti-Virus*, ACM SIGARCH Computer Architecture News, ACM Press New York, NY, USA. March 2005, pp, 90-98

[4] Nikita Borisov, Ian Goldberg, Eric Brewer. "Communication privacy: Off-the-record communication, or, why not to use PGP", *Proceedings of the 2004 ACM workshop on Privacy in theelectronic society,* Washington DC, USA. October 2004, pp. 77-84.

[5] Advance Key-logger by ELTIMA Software [Online]. Available: http://www.eltima.com/products/keylogger/

[6] Win-Spy by BC Computing Team [Online]. Available: http://www.win-spy.com/

[7] Muzammil M. Baig, Rana Naveed et all, "Anti-Hook Shield against the software Key-Loggers". *National Conference on Emerging Technologies*, SZABIST Karachi, December 18-19, 2004, pp.189-191.

[8] Matt Bishop. *"Computer Security: Art and Science"*. Addison-Wesley, 2002.

[9] Kevin Davis, "Saving users from themselves: creating

an effective student-oriented anti-virus intervention**",** *Proceedings of the 29th annual ACM SIGUCCS conference on User services*, Portland, Oregon, USA. October 2001, pp, 27-32.

[10] Grimes, Roger A.. *"Malicious Mobile Code"*. O'Reilly & Associates, 2001, pp. 190.

[11] Dieter Gollman. *"Computer Security"*. John. Wiley & Sons, Inc., 2000

[12] William Stallings. *"Network Security Essentials"*, (2nd Edition). Prentice Hall, 1999

[13] Abraham Silberschatz. *"Operating System Concepts*". John Wiley & Sons, 2001

[14] Gibson, Steve. *"The Anatomy of File Download Spyware"*. Gibson Research Corporation, .July 14, 2000.

[15] Invisible Key Logger [Online]. Available: http://www.invisiblekeylogger.com/ invisible-keylogger.html

[16] KeyGhost [Online]. Available: http://www.keyghost.com

[17] Lein Harn, Hung-Yu Lin, Shoubao Yang, "A software authentication system for the prevention of computer viruses**".** Proceedings of the 1992 ACM annual conference on Communications**,** Kansas City, Missouri, USA, April 1992, pp: 447 – 450.