2010 International Conference on Communications and Mobile Computing

Detecting Software Keyloggers with Dendritic Cell Algorithm

Jun Fu
Computer School
Wuhan University
Wuhan, China
Email: doctorfj@163.com

Yiwen Liang
Computer School
Wuhan University
Wuhan, China
Email: ywliang@whu.edu.cn

Chengyu Tan
Computer School
Wuhan University
Wuhan, China
Email: nadinetan@163.com

Xiaofei Xiong
Computer School
Wuhan University
Wuhan, China
Email: nanerzhi1987@gmail.com

***Abstract*—As a kind of invisible spyware that records user's keystrokes, software keyloggers have posed a great threat to user privacy and security. It is difficult to detect keyloggers because they run in a hidden mode. In this paper, an immune-inspired dendritic cell algorithm (DCA) was used to detect the existence of keyloggers on an infected host machine. The basis of the detection is facilitated through the correlation (including the timing relationships) between different behaviors such as keylogging, file access and network communication. The results of the experiments show that it is a successful technique for the detection of keyloggers without responding to normally running programs.**

***Keywords*-keylogger; Dendritic Cell Algorithm(DCA); Artificial Immune Systems(AISs); correlation; API function call**

I. Introduction

System security and privacy always have to face new confronts. In this never-ending combat, the latest challenge is keyloggers, a type of spyware which enables malicious users to harvest passwords and other confidential data, because of their rapid growth and wide spreading over the Internet. According to a report published in March 2007 by Kaspersky Lab [1], the number of keyloggers rose by more than 500% between January 2003 and July 2006. Meanwhile, keyloggers are becoming more diverse, sophisticated, evasive, and increasingly difficult to detect by anti-virus software and anti-keyloggers based on signature analysis [2]. Therefore, security experts are now focusing on run-time keylogger detection techniques that analyze API calls of a process to classify it as benign or malicious.

One research [3] disassembles all running processes searching for *SetWindowsHookEx* API calls used by some keyloggers. The problem of this method is that it is easy for keyloggers to evade the detection by using different methods to log keystrokes and false alarms are generated easily since legitimate applications also use this function to set hooks. To avoid these problems, another group [4] correlates different behaviors (logging keystrokes, accessing files and sending packets) in a specified time-window to detect keylogger processes. This method reduces the false positive rate to some extent. However, since the authors correlated keyboard events with the other two behaviors respectively and do not take into account the sequence of them, the method can not fully measure the correlation between different behaviors. In addition, specified time-window makes it possible for keyloggers to evade this detection [4].

The Dendritic Cell Algorithm (DCA) is an immune-inspired population based algorithm in artificial immune systems (AISs). It is based on an abstract model of the behavior of dendritic cells (DCs) which are natural intrusion detection agents of the human body [5]. As an algorithm, the DCA performs multi-sensor data fusion on a set of input signals, and these signals are correlated with potential 'suspects', leading to information which will state not only if an anomaly is detected, but in addition the culprit responsible for it. The ability of the DCA to correlate different behaviors, along with its population based feature which enables the usage of different time-window sizes, makes the DCA a good candidate solution for keylogger detection.

In this paper, considering the time sequence of behaviors exhibited by keyloggers, we apply the DCA to the detection of keyloggers based on the fact that keyloggers first intercept keystrokes of users, then save this information to log files and send it back to attackers. We focus on three types of keylogger behaviors: keylogging activity, file access and network communication. By monitoring selected behaviors represented by Windows API calls for running processes, we use the DCA to correlate these API calls to determine the presence of a keylogger infection.

II. Background

A. Software Keylogger Behaviors

Software keyloggers are applications that capture a computer user's keystrokes and then send this information back to a spying person [3].By Binding to free software, it is very easy to install keyloggers on victim's computers without user's awareness.

Unlike viruses and worms, the goal of keyloggers is generally not to cause damage or to spread to other systems. Instead, keylogger programs monitor the keystrokes and steal private information by capture what is done on a computer. Keyloggers capture keystrokes and save this information in hidden log files, then send it back to attackers. In this process, keyloggers have small footprint in terms of memory and processor utilization. Some of them even do not

978-0-7695-3989-8/10 $26.00 © 2010 IEEE
DOI 10.1109/CMC.2010.269

show up in the 'Task Manager' or in the list of processes. The log files are hard to distinguish from operating system files even when doing a directory listing of hidden files. In a word, they perform their task in strong disguise. However, in order to achieve their goal, they have to eventually execute the following malicious behaviors:

- First, monitor keystrokes of the user
- Second, leak the information captured (e.g., save the data to a file or transmit information to a remote host)

B. Keylogger Detection Techniques

There are only a few existing techiques for software keylogger detection. Most of these techniques use signature-based detection. The biggest disadvantage of this technique is that it has nothing to do against novel keyloggers [6].

Another category of detection method is behavior based detection. In this technique instead of looking for the specific file signature, the behavior of the application is scrutinized.

As keyloggers always use Windows hooks, Aslam [3] disassembles all running processes searching for *SetWindowsHookEx* used by some keyloggers to find keylogger processes. This method could discover keyloggers never seen before. However, it is easy for keylogger developers to evade this detection technique by using different methods to log the user activities other than using *SetWindowsHookEx*. Meanwhile, since legitimate applications also use this function to set hooks, this method inevitably has a high false positive rate [4]. In addition, disassembling all processes searching for *SetWindowsHookEx* is a tedious task.

Al-Hammadi [4] suggests that it is hard to detect keyloggers accurately with single behavior peformed by keyloggers and proposes a method to detect keylogging functions in bot program with corelations between different behaviors. This method monitors Windows API calls generated by keylogging, file access and network communication, and uses Spearman's Rank Correlation (SRC) [7] to correlate these function calls in a specified time-window. Rather than attempting to detect keylogger functions via single behavior that sets hooks [3], the author takes into consideration multiple behaviors of keyloggers. As a result, this technique has a relatively low false positive rate. However, SRC can only calculate the correlation between two variables. The author has only considered the correlation between keylogging and the other two behaviors respectively. And this method does not take into account the timing relationship between samples, so the methods can not fully measure the correlation between the three kinds of behaviors. In addition, specified time-window makes it possible for keyloggers to evade this detection.

C. Dendritic Cell Algorithm (DCA)

In this section, we present a brief introduction to DCA. For further information regarding the function of natural DCs and the details of the DCA can be found in [8].

The DCA is derived from an abstract model of DC biology resulting in a population based alogrithm, with each agent represented as a DC. Each cell has the capacity to collect input signals (PAMP, danger, safe) which can show the changes of the conditions of the monitored system and antigens who are responsible for the changes. The combination of the input signals forms cumulative output signals (CSM, semi-mature, mature) of DCs. The transformation from input to output signal per cell is performed using a simple weighted sum as:$O_j = \sum_{i=0}^{i=2} W_{ij} * S_i, \forall j$, where *W* is the weight matrix, *S* is the input signal vector, *O* is the output signal vector, *i* refers to the category of input signal, *j* refers to the category of output signal.The weight values are described in detail in [8].

DC spends time to collect signals and antigens. As the level of input signal experienced increases, the CSM output signal (O_1) also increases. Once CSM reaches a 'migration' threshold, the cell stops signal and antigen collection and is removed from the population for antigen presentation. In order to derive a context for the presented antigen, semi-mature output signal (O_2) and mature output signal (O_3) are compared in value. The context is termed as safe if O_2 is greater than O_3, and vice-versa. To keep the population static, the cell is replaced by a new one. Each DC is assigned a different migration threshold value, causing different cells sampling for different durations and experience different input signal combinations.

Each antigen is sampled multiple times so that is can appear in different contexts. In order to identify potentially malicious antigens, they are tagged with a mature context antigen coefficient, MAC [9]. The MAC value is calculated as: $MAC_i = \frac{N_{mi}}{\sum Ag}$, where i refers to the antigen type, N_{mi} refers to the number of times that antigen(Ag_i) has appeared in the mature context and $\sum Ag$ is the total number of antigens.

III. DCA for Keylogger Detection

A. Data Collection

Given that the host to be monitored was infected by a keylogger without user awareness, we implement a 'hook' program to monitor API calls generated by running processes in the host. The hooking program is based on modifying the process Import Address Table [10] to point to the replacement function instead of the original one. Thus, we were able to capture the function made by the keylogger. These function calls are as follows:

- **Keyboard State Functions**: *GetKeyboardState*, *GetAsyncKeyState*, *GetKeyNameText* and *keybd_event*.
- **File Access Functions**: *CreateFile*, *OpenFile*, *ReadFile* and *WriteFile*.
- **Communication Functions**: *socket*, *send*, *recv*, *sendto*, *recvfrom* and *IcmpSendEcho*.

These API calls are stored in a log file for further processing. Invoking these function call within specified time-window can represents a security threat to the system, but also may form part of legitimate usage. Therefore, an intelligent correlation method such as the DCA is required to determine if the invocations of such functions are indeed anomalous. Signals and antigens are vital input to the DCA. We next give our definitions of signals and antigens, and present our dendritic cell algorithm for keylogger detection.

B. Signals

Signals are a reflection of the state of the monitored system. A major function of the DC is processing signals of different categories. Five signals, namely one PAMP signal (PAMPs), two danger signals (DS) and two safe signals (SS), are used to define the state of the system. These signals are derived from the API calls captured, and then analyzed by the DCA, following a signal normalization process. The normalized signals range from 0 to 100 for the PAMP and DS with the SS having a reduced range, as suggested in Greensmith et al [11].

PAMP signal is a signature based signal. This signal is derived from the rate of invocation of keyboard status functions. Such function calls such as *GetKeyboardState* and *GetAsyncKeyState* are often used by keyloggers to record keystrokes. A large number of these function calls indicates the potential existence of the keylogger. Let the signal value be 100 (danger) when the rate exceeds N_{ph} and be 0 (not danger) when the rate is less than N_{pl} ($N_{pl} < N_{ph}$). We normalized the PAMP signal base on these values by applying linear scale between 0 and 100.

Danger signal is a measure of an attribute which increases in value to indicate an abnormality. Low values of this signal may not be anomalous.

DS-1 is derived from the time difference between two consecutive *WriteFile* function calls. As the keylogger has to save the keystrokes captured to the log files continuously, a small time difference between two writing file activities is observed. In contrast, normal application will have a higher value of time difference between writing activities. Let the signal value be 0 (not danger) when the time difference exceeds N_{d1h} and be 100 (danger) when the time difference is less than N_{d1l} ($N_{d1l} < N_{d1h}$). We normalized DS-1 base on these values by applying linear scale between 0 and 100.

DS-2 is derived from the correlation between different categories of function calls. Based on the behavioral characteristics of the keylogger, we generate this signal when the keylogger invokes file access or communication functions shortly after the invocations of keyboard status functions. The value of the signal lies on the sum of the number of the file access and communication function calls within specified time-window. Let the signal value be 100 (danger) when this sum exceeds N_{d2h}. DS-2 is normalized base on this value by applying linear scale between 0 and 100.

Safe signal is a confident indicator of normal, predictable or steady-state system behavior. This signal is used to counteract the effects of PAMP signals and danger signals.

SS-1 is derived from the time difference between two outgoing consecutive communication functions including *send*, *sendto* and *socket* functions. This is needed as keyloggers send information to the attackers after the keylogging activity. In normal situation, we expect to have a large time difference between two consecutive functions. In comparison, we expect to have a short period of this action when the keylogger sends information to the attacker. Let the signal value be 100 (not danger) when the time difference exceeds N_{s1h} and be 0 (danger) when the time difference is less than N_{s1l} ($N_{s1l} < N_{s1h}$). We normalized SS-1 base on these values by applying linear scale between 0 and 100.

SS-2 is derived from the small amount of the keyboard status function calls within a specified time-window. As legitimate applications such as Notepad or Wordpad invoke much fewer keyboard status functions than keyloggers. So, small amount of invocations is considered to be safe in the host. Let the signal value be 0 (danger) when this amount exceeds N_{s2l}. SS-2 is normalized base on this value by applying linear scale between 0 and 100.

C. Antigen

Antigens are potential culprits responsible for any observed changes in the status of the system. As any process executed one of the selected API functions explained in the beginning of this section, the process id (PID) which causes the calls and thus generates signals is defined as antigens. Observing which processes are active when signal is generated, we could find the existing keylogger in the system.

Processing signals, the DC stores PID which generates signals and give the context to the stored PID when the DC changes its state.

IV. Experiments

The aim of our experiments is to apply the DCA to the detection of the keyloggers. For the purpose of experimentation, spybot [12] is used as the keylogger to be identified. The spybot is a typical keylogger that is usually hidden in the infected host to steal user input information and save it to the log files. When command received, it will continue to send the information back to the attackers in real time.

As the spybot communicates with the attacker through IRC protocol [13], all experiments are performed in a small IRC network. The IRC network consists of two machines. One is infected with spybot, running Notepad and IRC client (mirc [14]) as normal applications. The other is IRC server running mirc only. The attacker sends commands to the spybot through mirc connected to the server, and the spybot sends the intercepted information back to it. Two machines run under a Windows XP SP3 with a

2.2Ghz processor and 512MB RAM. The parameters in the DCA were set as follows: $numberofcells = 10$, $N_{ph} = 1000(times/s)$, $N_{pl} = 100(times/s)$, $N_{d1l} = 1000(ms)$, $N_{d1h} = 3000(ms)$, $N_{d2h} = 8(times/s)$, $N_{s1l} = 1000(ms)$, $N_{s1h} = 2500(ms)$, $N_{s2l} = 30(times/s)$. All parameters are values of experience. We used same weight values as in [9].

We have performed two experiments, each experiment has two scenarios. The time of each scenario is 600 seconds. Without any operations in the first 60 seconds, we use Notepad and mirc to input keystrokes for 180 seconds respectively with an interval of 60 seconds. We have no operations in the final 120 seconds. Each scenario is repeated for ten times.

The aim of the first experiment (E1) is to verify our detection method when the spybot shows only two behaviors (keystroke logging and file access) because of the absence of keylogging commands. We monitor two scenarios of typing. In the first scenario (E1.1), we type short sentences while in the second scenario (E1.2), we type long sentences. The sentence ends with [Enter] key. By monitoring two typing scenarios, we are able to show the effect of different user's activity on our detection scheme.

The aim of the second experiment (E2) is to verify our detection method when the spybot shows three behaviors (keyboard recording, file access and network communication) after receiving keylogging commands. The same procedure is taken as in E1 where we have two scenarios of typing: short sentences (E2.1) and long sentences (E2.2).

A. Results

We analyze the results of the experiments described above. We first give a look at the function calls we intercepted in every scenario. For all scenarios, the x-axis represents time in seconds while the y-axis represents the normalized value of functions. The normalized function call frequency values represent the total value we get per second divided by the maximum value of the whole period.

In E1, although the attacker did not send keylogging command to the spybot, we observed a large amount of keyboard status and file access API calls in both E1.1 (as in fig.1) and E1.2 (The figure is not shown because of the similarity to fig.1)when sentences were typed in Notepad (61-240 second) or mirc (301-480 second).The data is from a randomly selected experiment because the data from the repeated experiments produce a small variation. We also noticed that the value of file access function calls in E1.1 is higher than the value in E1.2. In addition, we discovered that there is some network function calls every specified time intervals. This is due to the regular communication between the spybot program and the IRC server.

In E2, the attacker sended keylogging command to the spybot, we observed obvious malicious activities exhibited by the spybot. As shown in fig. 2, when sentences were typed in Notepad (61-240 second) or mirc (301-480 second), we found a large amount of keyboard status function calls and an increase in file access and communication function calls in both E2.1 and E2.2 (The figure is not shown because of the similarity to fig. 2). We also noticed that the value of file access and communication function calls in E2.1 is higher than the value in E2.2.

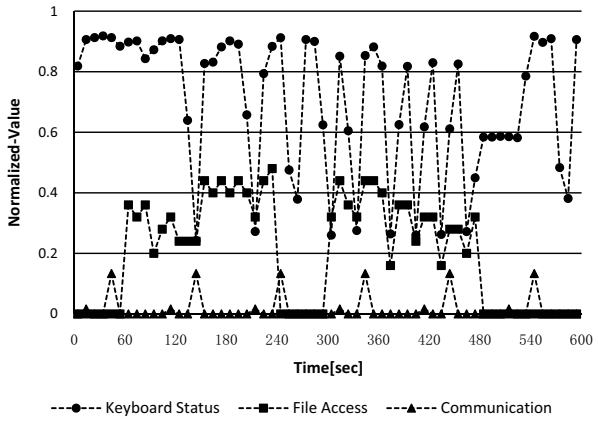

Figure 1. API functions invoke by spybot in E1.1

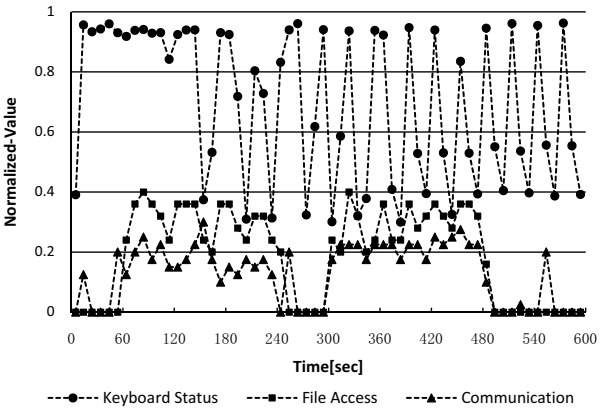

Figure 2. API functions invoke by spybot in E2.1

Table IV-A gives the results from the DCA for the detection of the keylogger. The values in the table are the mean and variance in five repeated experiments.

A threshold(T) is applied to MAC to make the final classification decision. The antigens with MAC higher than T are termed malicious, andvice-versa. Because the dataset used contains one malicious instance and two benign instances, we can define $T = \frac{1}{1+2}$. So the process with $MAC > 0.33$ is considered to be the keylogger process, then the content in bold in the table is identified as the keylogger process by the detection system. From the table, we observed that the spybot in each scenario is detected with different detection confidence. $MAC \geq 0.67$ is high confidence detection, and $0.33 < MAC < 0.67$ is low confidence detection.

Table I
RESULTS FROM THE EXPERIMENTS

Scenario	Process Name	The Number of Antigens	MAC	Detection Confidence
E1.1	**spybot**	**11663(19.2)**	**0.701(0.010)**	high
	notepad	1676(0.7)	0.100(0.003)	
	mirc	2133(1.5)	0.112(0.006)	
E1.2	**spybot**	**6797(31.5)**	**0.489(0.026)**	low
	notepad	1872(0.9)	0.126(0.012)	
	mirc	2154(1.3)	0.099(0.023)	
E2.1	**spybot**	**12788(17.3)**	**0.724(0.010)**	high
	notepad	1851(0.6)	0.100(0.004)	
	mirc	1872(3.1)	0.102(0.005)	
E2.2	**spybot**	**7786(22.1)**	**0.467(0.059)**	low
	notepad	1816(0.7)	0.099(0.026)	
	mirc	1975(5.6)	0.074(0.019)	

B. Discussions

The results of the experiments show that the DCA could differentiate the keylogger process from the normal processes in the infected host from the behavior point of the view. In addition, the time-window of the correlation is not specified as the DCs have different migration thresholds, causing different cells sampling for different durations and experience different input signal combinations.This makes it difficult for the keylogger to evade the detection. It could be concluded that the DCA has a high detection rate and low false alarm rate when applied to the detection of keyloggers.

From the low detection confidence in E1.2 and E2.2 in Table 2, we noticed that the input mode of the users has an effect on the detection confidence. The detection confidence decreases when long sentences are encountered. This may due to the spybot obscure malicious activities caused by the low rate of [Enter] key in both E1.2 and E2.2. The analysis of the spybot demonstrated that the spybot starts its file access and communication activities when [Enter] key is encountered. In future, the obscurity could be alleviated by amplifying the characteristics of the keylogger behaviors.

In addition, parameter values have an impact on the results of our approach. The parameter values we set are derived from the experience of the repeated experiments. The future work is to discover the principles of how to set the parameter values.

V. CONCLUSION

In this paper, considering the time sequence of behaviors exhibited by keyloggers, we apply the DCA to the detection of keyloggers based on correlations between keylogging, file access and network communication behaviors. The results of the experiments show that the method proposed in this paper could differentiate the running keylogger process from the normal processes with a high detection rate and a low false alarm rate. The future work is to use the results of these experiments to further our understanding of the DCA, to ultimately enhance the performance of this immune-inspired detection algorithm.

VI. ACKNOWLEDGMENTS

This work is supported by the Research Project No.A1420080183 from Ministry of Education of People's Republic of China.

REFERENCES

[1] "Keyloggers become cyber criminal tool of choice," http://www.v3.co.uk/vnunet/news/2186806/keyloggers-become-cyber.

[2] M. Baig and W. Mahmood, "A Robust Technique of Anti Key-Logging using Key-Logging Mechanism," in *Inaugural IEEE-IES Digital EcoSystems and Technologies Conference, 2007. DEST'07*, 2007, pp. 314–318.

[3] M. Aslam, R. Idrees, M. Baig, and M. Arshad, "Anti-Hook Shield against the Software Key Loggers," in *Proc. of Nat. Conf. of Emerging Technologies*. Citeseer, 2004, pp. 189–191.

[4] Y. Al-Hammadi and U. Aickelin, "Detecting Bots Based on Keylogging Activities," in *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*. Citeseer, 2008, pp. 896–902.

[5] J. Twycross and U. Aickelin, "Biological inspiration for artificial immune systems," *Lecture Notes in Computer Science*, vol. 4628, p. 300, 2007.

[6] M. Krasnostup and D. Kudin, "Anti-Spyware: Efficiency of the Means of Defense," http://www.keylogger.org/articles/mykola-krasnostup/anti-spyware-efficiency-of-the-means-of-defense-13.html.

[7] G. Bancroft and G. O'Sullivan, *Maths and statistics for accounting and business studies, 2nd ed*. McGraw-Hill Book Company (UK) Limited, 1988.

[8] J. Greensmith, "The dendritic cell algorithm," Ph.D. dissertation, School of Computer Science, University Of Nottingham, 2007.

[9] Y. Al-Hammadi, U. Aickelin, and J. Greensmith, "Dca for bot detection," in *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI) Hong Kong*, 2008, pp. 1807–1816.

[10] H. Father, "Hooking Windows API-Technics of hooking API functions on Windows," *Assembly-Programming-Journal*, vol. 2, no. 2, 2004.

[11] J. Greensmith, U. Aickelin, and G. Tedesco, "Information fusion for anomaly detection with the dendritic cell algorithm," *Information Fusion*, vol. 11, no. 1, pp. 21–34.

[12] P. Barford and V. Yegneswaran, "An inside look at botnets," *Malware Detection, ser. Advances in Information Security. Springer-Verlag*, pp. 171–191, 2006.

[13] C. Kalt, "Internet relay chat: Architecture," RFC 2810, April 2000, Tech. Rep.

[14] "mIRC client application," http://www.mirc.com.