Malware Implementation/Detection/Prevention

Anonymous

ABSTRACT

With the exponential increase over the years, there is never a day without the cyber attacks happening. There has been an increasing awareness among the professionals. It is important to understand that the lack of secure coding practices is one of the primary reasons for these attacks. The attackers exploits the vulnerabilities, bugs or errors to take control of the system. The vulnerabilities are caused due to the human coding errors, design flaws, etc and identifying these errors help the developers to eradicate the vulnerabilities and secure the system. In order to create a better defense, it is important to think like a black hat hacker and a white defender. The objective of this project is to implement a malware to infect a target machine and perform detection and prevention techniques to eliminate and mitigate the vulnerabilities of the system. The API based keylogger is implemented using the windows hooking mechanisms. During the detection phase, new tools like Process Hacker, TCPView and NMap are employed to detect the malware presence in the system.

Keywords: Malware, Keylogger, Windows Hooks.

1. OVERVIEW

Malware is intended to damage a computer's operation or to steal the confidential information of a user or even obtain access to users' private computer system without their knowledge. Malwares can be in the form of a code, a script or may even be some other software. In most of the cases, a malware is placed inside non-malicious files. It is basically a collective term to refer all kinds of threats such as viruses, spywares, worms or some malicious programs like keyloggers, rootkits, etc.

The main objective of this paper is to design and implement a malware to infect a target machine and design a detection and prevention model which will detect any malware kind. This paper focuses on keylogger which is our major focus of implementation in the project. Keylogger, otherwise called as keystroke logging or keyboard capturing, is a technique of recording the users' activity of keystrokes and storing the information in a log file. The keyloggers could be hardware-based or software-based. Software-based keyloggers are computer programs while the hardware-based keyloggers act on the hardware level of the system. Unlike other malwares, keyloggers pose no threat to the system. However, it could be a threat to the user's sensitive information. Many passwords, account numbers, PINs and credit card information have been stolen through the keyloggers, thus violating the confidentiality of a secure system. Although it is used mostly by malicious users, the keyloggers can be legitimately used for several surveillance and monitoring applications.

There are many categories of software-based keyloggers like Hypervisor-based, kernel-based, API-based, Form grabbing based, etc. We found three research papers for keyloggers. The research paper by Yang et al [6] explains the implementation of the API-based keylogger using the Windows API. The research paper by Ladakis et al [5] discusses how graphic card can be used as an alternative to the keyloggers in order to record all the user keystrokes and store them in the memory space of GPU whereas the research paper by Al-Hammadi et al [4] explains an algorithm which makes use of function calls in order to detect bots using the keylogging component of the user. Out of these three research papers, we decided to focus on the research paper by Yang et al [6] which is based on the API-based keylogger, which we will be implementing in our project. This research paper by Yang et al [6] explains the implementation of the API-based keylogger through the hooking mechanism. The hooking mechanism intercepts actions like keystrokes, mouse clicks, etc, allows to modify them and can interrupt them in the operating system level before it reaches the application. Here, the Windows API, informally referred to as WinAPI, is used for implementing the keylogger. This paper by Yang et al [6] also explains about Win32 API, the API version for modern Windows. The Win32 API offers several hook options for events like keyboard stroke, mouse clicks, etc. The hooks that are of interest for the keylogger are: `WH_KEYBOARD` [3] and `WH_KEYBOARD_LL` [3] which monitors the keyboard input events. The implementation of keylogger uses hook functions such as SetWindowsHookEx() [3] to install a hook procedure, GetAsyncKeyState() [2] to check whether the keyup or keydown event occurred, GetKeyNameText() gets the name of the key pressed, CallNextHookEx() [2] to send the messages to the next hook and UnHookWindowsHookEx() [3] to uninstall the hook. The keylogger program will store the users' keystrokes in the form of a text file. Thus, our API-based keylogger records the keystrokes and mouse clicks of the users thus violating the confidentiality of user data.

The second phase of this paper will also include the analysis of the keylogger in addition to implementation of the keylogger and will propose a generalized detection and prevention mechanism for any malware. The other sections of this paper will be updated as the project progresses.

2. DESIGN CONSIDERATIONS

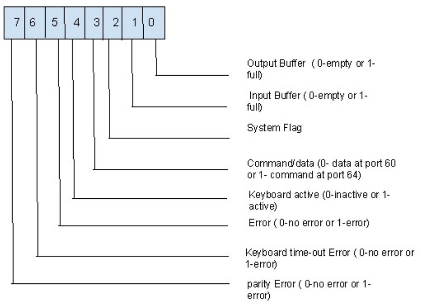
7 6 5 4 3 2 1 0
Output Buffer (0-empty or 1-full)
Input Buffer (0-empty or 1-full)
System Flag
Command/data (0- data at port 60 or 1- command at port 64)
Keyboard active (0-inactive or 1-active)
Error (0-no error or 1-error)
Keyboard time-out Error (0-no error or 1-error)
parity Error (0-no error or 1-error)

Figure 1: On-board microcontroller status byte Port 64

The better understanding of the internal working of the keyboard and the hooking mechanism in the windows platform is an essential key and makes it easier to understand the mechanism for implementing keylogger. The readers should understand that the hackers/malicious users are well-aware of these mechanisms as well. This section provides a detailed explanation of the above mentioned.

The computer keyboard is one of the primary interactive input devices. The keyboard is a hardware device that is connected to the computer through the USB port or PS/2 port. There are two microcontrollers associated with the keyboard: one is the on-board microcontroller which is on the PC motherboard and the other is the keyboard microcontroller in the keyboard. The keyboard is itself a small computing system with 8042 microcontroller which acts as an interface between the keyboard and the computer. The 8042 microcontroller processes the keystrokes whenever a key is pressed on the keyboard. The keystrokes are never missed by the controllers as the processing happens in parallel with the other processes. Each key on the keyboard are assigned two scan codes. The scan codes are not the ASCII codes. Whenever a key is pressed a down code is generated and when the key is released an upcode is generated. These scan codes are linked to a keyboard matrix map. The generated scan codes are translated by a controller, send it to the input/output port 60 and generates a hardware interrupt service. The keyboard interrupt service reads the translated code at the input port, gets the status of the key pressed and writes the keystroke by sending the write command to the controller. The on-board microcontroller takes the control now. The input/output process of the on-board microcontroller takes place at the I/O port 64. The 0th and 1st provides handshaking between the ports 60 and 64. Initially, the port 64 0th bit is set to zero; once the port 60 notifies the port 64 saying that the scan code is ready for reading the 0th bit is set to one. The read keystroke data are displayed to the applications by the operating system drivers i.e keyboard drivers. The figure 1 shows the on-board microcontroller status byte at the post 64.

The keyboard gets access through Microsoft Win32 API by using the Raw Input Thread (RIT). The RIT is created whenever the system is booted and gains access to the keyboard driver. Whenever the key is pressed or released, a hardware interrupt is generated by the keyboard controller. The keyboard events are put in the hardware input queue and is converted into windows messages and are placed in the virtualized input queue (VIQ). The windows explorer then creates threads for each of the application (e.g a thread for MS word, Notepad, etc). These threads are then bonded to the raw input thread. At a given time, only one thread could be associated with the RIT. So, whenever a new thread tries to connect to the RIT, the previous thread gets unhooked from RIT. Briefly, whenever a user presses a key on the keyboard, the request is placed on the system hardware input queue (SHIQ), RIT is activated, the keyboard input event is then transformed into a message and the corresponding key is placed on the application message queue.

The mechanism used by the keylogger to intercept the keyboard events using functions in Microsoft Windows is called hooking. When an application is hooked, the messages passing from the keyboard to the operating system is sent to the hooks before it reaches the target application. The hooked application then copy, modify or delete the received message. The hook program is pre-started before any application is initiated, made available at a separate dynamic library (dll) and is injected to all the processes. There are two types of hooks: global hook and local hook. The global hooks can monitor the whole system wide messages and the local hooks can monitor a targeted application messages. Microsoft Windows has 14 different kinds of hooks. The few of the hooks are `WH_KEYBOARD_LL` [3], `WH_MOUSE_LL` [3], `WH_GETMESSAGE_LL` [3], etc. The Win32 API SetWindowsHookEx function installs the hook and UnhookWindowsHookEx function uninstall the hook. The hooks follows LIFO order in processing the processes. The CallNextHookEx function is used to pass the messages to the next hooks. After passing it to the next hook, the messages can be sent to the user interface of an application and the hooks can be uninstalled. The parameters required for each of the hook functions are explained in the section 4.

3. ARCHITECTURE

This sections gives an overview of the architecture of the keylogger employed in our project. This paper talks about the keylogger implementation using windows hooking mechanism through the Win32 API. When a key is pressed/released, the keystrokes are processed by the microcontrollers and the process is taken care of by the operating system internally as explained in the section 2. The hooking mechanism processes all the keyboards events happening in the computer system. Before it is been sent to the targeted user application, the virtual code of the keys and the key characters pressed are first recorded and stored in a textfile. The textfile is then stored at a location which would be unknown to the user. Once this is done, the keystroke data is sent to

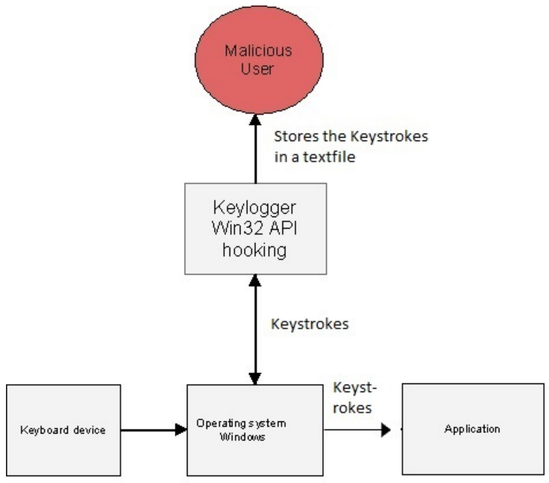

Figure 2: Architecture of the Keylogger Project

the target application and is displayed in the application for the user to view. The keylogger hides its presence by taking control of the interfaces. The figure 2 gives an overall simple architecture view of our project. The API functions used to implement the hooking mechanism are clearly explained in the section 4.

4. IMPLEMENTATION

In this project, we will be implementing the key logger using the windows hooking mechanism. We first started with installing the Windows 7 Base in the rlesvcloud as our virtual machine. For writing our code we used the Qt Creator IDE and used C++ programming language. The hooking mechanism, has a filter function which can make changes to the events or the messages sent even before they actually reach the application. We have used the SetWindowsHookEx function [2] in order to install the hook on the keyboard. This function takes in four parameters: idHook (specifies the type of hook we have used i.e. `WH_KEYBOARD_LL`) [3], lpfn (a pointer to the hook procedure), hmod (specifies the pointer to the DLL which contains the hook) and dwThreadid (number of existing threads that the hook procedure is related with).

After the hook is successfully installed, it is placed on the top of the hook list by the SetWindowsHookEx [2] function. Once an event or a message reaches our hook, we invoke the CallNextHookEx [2] function in order to pass the event to the next hook which then can be used by its respective application or program. CallNextHookEx [2] function has four parameters: hhk (this is an optional parameter), nCode (this is used by the next hook to process the hook information), wParam (it is a windows message which gives information whether a key is pressed or released) and lParam (this gives the information about the actual key that is pressed). We have used the GetAsyncKeyState [2] function in order to determine whether a key is pressed or not, at the time, the function has been called. We have also used the GetKeyNameText [2] function to get the string representing the name of the key and MapVirtualKey function which converts the virtual-key code into the value of the characters or converts the scan code into the virtual key code.

Finally, after everything has been done, we use the UnhookWindowsHookEx function [2] to uninstall the hook from the system. This hooking mechanism helps us in recording the keys pressed by the user and enables us to record the users' activity and thereby storing their confidential information without their knowledge. In order to store all the keystrokes that have been recorded by our key logger we created a textfile in our virtual machine, whose location would not be known to the user. We open the textfile created, such that every time any key has been pressed and the key code values obtained are added to that textfile. In order to differentiate the user activity every time he logs in and presses the keys, we append the keystrokes at the end of the previous output in the textfile. So in this way the textfile records all the keys pressed by the user.

The c++ program file is converted to the .exe format. The .exe file is put in the Windows startup folder in order to ensure that the keylogger runs automatically once the windows system boots. Initially, the keylogger.exe file displays the results on a console window which is visible to the users. To eliminate this, we used a tool called Hstart - Hidden Start [1], which hides the console of the window from the users. This ensures that the users cannot see the output of the keylogger running on their system. This keylogger works behind any application including the desktop on the users' system.

5. ETHICAL ISSUES

One of the most important ethical issue which our project violates is confidentiality of a secure system. This is because the keylogger can be used to get important information of the users like password, PINs, credit card information without their knowledge. So this in turn also results in an unauthorized access of the various personal information of the users. This violates the privacy of their personal data and the integrity of the system. Moreover it also makes the personal data available to the unauthorized people without the data owner knowing about it.

6. LESSONS LEARNED

During the investigation phase our log file was accessible since it showed up in the modified files list in the process hacker tool. So it is important to make sure that the log file always stays hidden so that the tools cannot access it. Also during the investigation on other teams, we noticed that their malware was consuming a lot of CPU resources. So it is necessary to ensure that the malware does not consume a lot of resources. We also got to work with new tools like Process hacker, Wireshark, NMap and TCPView. Overall this project was a good learning experience as we learnt the steps involved in developing the malware, breaking into the system and detect the malware presence using the existing tools.

7. CURRENT STATUS & FUTURE WORK

We have currently implemented the keylogger using the Windows hooking mechanism, which directly sends information about the keys pressed, to the console of the Qt Creator IDE. In order to keep the information about the key pressed hidden from the user, we created a textfile in our virtual machine, where the location of the textfile would not be known to the user and store all the key strokes in that file. Also we hide the Qt console, by using a tool called Hstart [1] so that the user does not get any idea about the output of the keylogger running in his system. Finally an executable of our program is created, which runs automatically when the virtual machine reboots. So the user would not know about the keylogger being run in the background. The future work can include designing an email extension to our existing model so that the textfile can also be directly emailed to the attacker.

8. REFERENCES

[1] Hstart tool. `http://www.ntwind.com/software/hstart.html`. Accessed: 2015-10-31.

[2] Keylogger tutorial. `http://www.secsavvy.com/malware/keylogger/keylogger-tutorial`. Accessed: 2015-10-30.

[3] Msdn document. `https://msdn.microsoft.com/en-us/library/windows/desktop/ms632589(v=vs.85).aspx`. Accessed: 2015-10-27.

[4] Y. Al-Hammadi and U. Aickelin. Detecting bots based on keylogging activities. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 896–902. IEEE, 2008.

[5] E. Ladakis, L. Koromilas, G. Vasiliadis, M. Polychronakis, and S. Ioannidis. You can type, but you can't hide: A stealthy gpu-based keylogger. In *Proceedings of the 6th European Workshop on System Security (EuroSec)*, 2013.

[6] G. Yang, H. Gao, and W. Kang. Theory, implementation and protection of keylogger analysis the method of keylogger using a live example. In *International Conference on Measurement and Control Engineering 2nd (ICMCE 2011)*. ASME Press, 2011.