

Anti-Hook Shield against the Software Key Loggers

Muhammad Aslam, Rana Naveed Idrees, Mirza Muzammil Baig, and Muhammad Asif Arshad
{aslam, naveed, muzammil, asif}@kics.edu.pk
Al-Khawarizmi Institute of Computer Science, University of Engineering and Technology
Lahore, Pakistan.

Abstract:- *Software Key logger is a stealth surveillance application, which is used to keep record of user activities on the computer in various ways like keyboard logging, screen logging, mouse logging and voice logging completely undetected to ANY user because it is designed to capture what is done on a PC. They have gained so much power in their execution and have become a serious threat to the privacy and security of a computer. Windows hooks play a major role in the development of software key loggers. We can guard our privacy by anti-hook mechanism, which has been devised for the sure shot detection of both known and unknown key loggers, currently in use or being developed at the present moment. Anti-hook technique is based on the fact that each processes either hidden or on display uses hooks APIs for the purpose of hooking. So if we become able to scan all the processes and static executables and DLLs and detect the suspicious processes or files, which uses hooks. Then we can get the complete detail about that particular file or process. We can also terminate its execution or existence to secure the system. This paper focuses the anti-hook technique by keeping in view the Key loggers development process so that personal privacy and security can be ensured.*

Keywords: *Key logger, Anti-Hook technique, Privacy and Security, Windows Hooks*

1. INTRODUCTION

Software based key logger is common component of Trojan horses that are often installed by gaining physical access to the computer or by downloaded programs. Their small footprint in terms of memory and processor utilization makes them practically untraceable. Key loggers can email or ftp the file containing keystrokes back to a spying person. Such key loggers often operate quietly in the background and capture whatever the user types on the keyboard, so all the keystrokes are stored in a well-hidden file.

This paper examines software key logger development procedure and anti-hook technique implementation to defeat such powerful and dangerous key loggers for making the user secure from this extravagant threat.

2. SOFTWARE KEYLOGGERS

A variety of keyloggers can be found by searching for “keylogger” using a search engine. Software key loggers can also be downloaded for free from any keylogger forum. For this project we tried both commercial as well as free software keyloggers. A software keylogger can be

installed on a machine with Administrator privileges. They come in various forms. A keylogger can be an executable. It can be a device driver that replaces the existing I/O driver with embedded key logging functionality. Most commonly, key loggers are written in C/C++ using Windows hooks.

We tested many known key loggers. Each of them worked differently but the end results were same. They logged keystrokes and mouse clicks and wrote them to a file. They had the option of encrypting and decrypting the log files and the option of sending the file to a destination across the Internet. Some of them even did not show up in the ‘Task Manager’ or in the list of processes. The key loggers log files are mostly hidden. These log files are hard to distinguish from operating system files even when doing a directory listing of hidden files.

3. KEY LOGGERS DEVELOPMENT PROCESS

Every key logger is consisting of two files: a DLL which does all the work and an EXE which loads the DLL and sets the hook. Therefore when you deploy the hooker on a system, two such files must be present in the same directory. Where as window’s hooks play a major role in their development. So let us look them in detail because their thorough understanding is a necessary to digest key logging as well as anti-hook techniques.

4. WINDOWS HOOKS

A Windows hook is the core of key loggers. A hook is a point in the system message-handling mechanism where an application can install a procedure to intercept message traffic before it reaches a target Window procedure.

A function can intercept events before they reach an application through this mechanism. The function can act on events, modify or discard them. Functions which receive the events are called Filter Functions; every Filter Function is classified by its type. Hooks provide powerful capabilities: Process or modify every message; Record or play back keyboard and mouse events; Prevent another filter from being called; and many more capabilities.

Generally, there are two types of hooks: System-wide, and Thread-specific. The System-wide hook is used for filtering messages of all applications (IE: when writing a key logger). And the Thread-specific hook is used for filtering messages of a specific thread. System-wide keyboard hook is used to develop key loggers. To set a System-wide hook we need a DLL.

The reason we need a DLL for a System-wide hook is because we want the Filter Function to be in any application address space. So when you set the hook message filter function which lies in the .dll , Windows maps the .dll automatically into all applications' address space. Thus we get our filter function called for every process! Therefore when we dynamic linking the hook which is in a .DLL it becomes a System-wide hook (of course it depends on the type of Filter Function too). A hook procedure has the following prototype.

```
LRESULT CALLBACK HookProc(
    int nCode , /* specifies the action to
                be performed */
    WPARAM wParam , /* parameter
                    depending on nCode*/
    LPARAM lParam /* parameter
                  depending on nCode */);
```

A hook chain is a list of pointers to hook procedures. When a message occurs that is associated with a particular type of hook, the system passes the message to each hook procedure referenced in the hook chain, one after the other. A hook procedure can monitor or modify a message passing through a hook chain. It can also prevent the message from reaching the next hook procedure or the target window procedure. The SetWindowsHookEx function installs an application-defined hook procedure at the beginning of the hook chain. It has the following function prototype:

```
HHOOK SetWindowsHookEx(
    int idHook , // specifies the hook
    HOOKPROC lpfn , /* pointer to hook
                    procedure*/
    HINSTANCE hMod , /* pointer to dll
                    containing the hook procedure*/
    DWORD dwThreadId /* identifier of associated
                    thread*/
);
```

5. HIDING THE KEY LOGGER

There are many ways to hide a running key logger from showing up in the task manager or the list of processors. To hide it from process viewer, we need to capture process ID from the process queue to make the application stealth surveillance. We have found that keyloggers are practically impossible to track once installed. Because they make themselves completely invisible that even the expert user fails to detect it. Here is one way to hide it by opening a hidden window on startup and setting all the required parameters. The following code hides the key logger from application level.

```
WNDCLASSEX wincl;
wincl.hInstance = hInstance;
wincl.lpszClassName = name;
wincl.lpfnWndProc = WndProc;
/*Make all window properties zero. This will
make the window invisible.*/
wincl.style = 0; wincl.cbSize =
sizeof(WNDCLASSEX);
wincl.hIcon = NULL;
wincl.hIconSm = NULL;
wincl.hCursor = NULL; wincl.lpszMenuName =
NULL; wincl.cbClsExtra = 0;
wincl.cbWndExtra = 0; wincl.hbrBackground =
0; wincl.lpszMenuName = NULL;
if(!RegisterClassEx(&wincl)) return 0;
/* Make all display parameters zero. This will
make it invisible from the taskbar*/
hwnd=CreateWindowEx(0,(LPCTSTR)name,"",
0,0,0,0,HWND_DESKTOP,NULL,
hInstance,NULL);
ShowWindow(hwnd,SW_HIDE);
```

6. ANTI KEY LOGGERS

Anti-key loggers are software that purports to detect key loggers. We installed and tested a lot of anti-key loggers for survey purpose. The anti-key loggers did not detect some of the software key loggers. The reason is that there are many ways for key loggers to work and hide themselves. Anti-keyloggers that detect many key loggers have a very high rate of false positives. These anti-key loggers monitor programs using Windows hooks, and hooks are legitimately used by many functions. So if we become able to find out the installed hook of a particular process. Then by getting some key information of the suspicious process, we can catch the culprit.

There are many techniques and approaches to implement the anti-key logger as we can smell the key loggers in different ways e.g. we can see the continuous file write on hard drive to judge the existence of key logger. But this approach is not suitable in certain scenarios e.g. key logger may not use hard drive for log file and using ftp access for its purpose. Anti-hook technique detects the key logger at the very basic layer of its design and it produces great efficiency in detecting them in many situations. So we can say it is the best available technique for the development of deadly anti-key loggers in response to deadly key loggers of this era.

7. ANTI HOOK TECHNIQUE

Anti-Hook is a mechanism through which we find out all the processes that use hooks. We become able to highlight all those processes and their detail through this technique, whether those processes are visible or invisible at any level of the application. Basically in this technique, we scan the system thoroughly in different scenarios and check for those processes, which are using hooks.

In this approach, we enumerate all the hidden as well as visible processes by index from task manger and then we enumerate the dynamic link libraries of each and every process. In the meanwhile, we obtain the name and other details of process

We are well aware of the fact that SetWindowHookEx command is used to install the hook and this API lies under the banner of USER32.LIB. So we disassemble all the running processes in search of SetWindowsHookEx. It is obvious that if the process or it's dynamic link libraries use this API. It shows the installation of hook. We first put a check of USER32.LIB on each and every process and its DLLs to filter the search from a pool of processes then we check USER32.LIB of a specific EXE/DLL for SetWindowsHookEx. If we find positive result then we post true message The goal of anti-hook technique is to find out SetWindowHookEx command in all type of processes and their respective dynamic link libraries. Anti-hook technique sends all information in the form of its output to an anti- key logger, where the user witnesses all the work that is done by Anti-Hook. The diagram on right shows the sequence of the anti-hook technique.

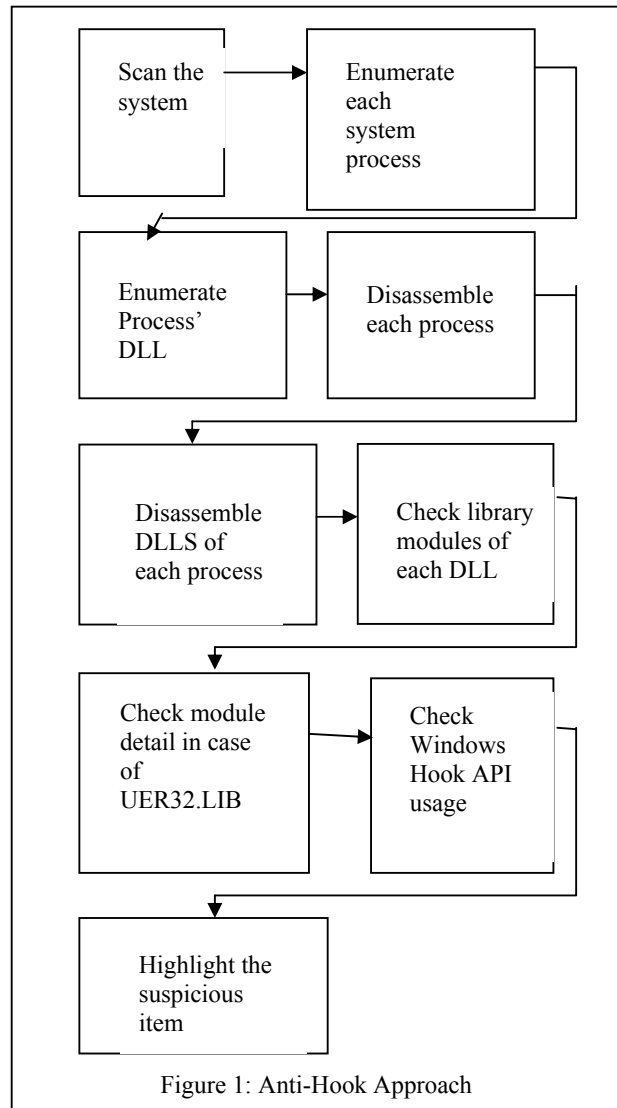
8. CONCLUSIONS

Key loggers are simple to write and simple to install. They are easily acquired by browsing the Internet or can be purchased at a modest price. The usage of keylogger can not be banned because some situations, we find it very useful as many administrators in some organizations, and some parents, might find some benefits. But privacy advocates may find no valid use for a keylogger. Those whose strokes are surreptitiously logged may be angered by the invasion of privacy. Anyhow due to the fact that it is not the gun but the person holding the gun who matters, so we should adopt security practices.

Security and privacy will always remain the biggest concern of the user. An Anti-key logger will be able to detect and terminate any suspicious process, with the help of anti-hook mechanism, which is able to record and steal sensitive information, passwords, logins, PINs (Personal Identification Numbers) etc. Anti-key loggers have become the need of the hour as it provides us protection. There is endless war between spy and anti-spy applications and we should prevent us only by keeping ourselves update with the advances in the field of security.

REFERENCES

[1] Dieter Gollmann. *Computer Security*. John. Wiley & Sons, Inc., 2000



- [2] Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2002
- [3] Grimes, Roger A.. *Malicious Mobile Code*. O'Reilly & Associates, 2001, (p. 190).
- [4] William Stallings. *Network Security Essentials (2nd Edition)*. Prentice Hall, 1999
- [5] Abraham Silberschatz. *Operating System Concepts*. John Wiley & Sons, 2001
- [6] Gibson, Steve. *The Anatomy of File Download Spyware*. Gibson Research Corporation, .July 14, 2000.