ISSN:0975-9646

Hypervisor based Mitigation Technique for Keylogger Spyware Attacks

C.Santwana
Dept. of Information Technology,SRM University
Chennai, India

K. Sai Aditya
Dept of Information Technology,SRM University
Chennai, India

Dr. S.Magesh
Assoc. Professor
Dept. of Information Technology,SRM University
Chennai, India

***Abstract*—Key loggers have been posing a threat to the confidentiality of sensitive information such as passwords, credit card numbers..Etc. Keyloggers are very difficult to identify as they either work in stealth mode or pose themselves as legitimate programs to the system. There are several techniques for overcoming the menace of key loggers, but none is sufficient on their own. There needs to be a combination of several techniques to effectively eliminate the problem. We examine various techniques for detection of key logger attacks and concentrate upon the password input isolation using hypervisor in order to mitigate their effect as much as possible.**

***Index Terms*—Key loggers, Keyloggers, Hypervisor, trusted computing base, malware, data stealing**

I. INTRODUCTION

Most of the commerce activities of present day world are transforming with jet speed into internet applications. These activities include banking, fund transfers, credit and debit card transactions of e-shopping sites...etc. But the computers of modern days are in a highly insecure condition, owing to several threats arising from adversaries upon the internet. Key loggers are one such grave threat to the privacy of sensitive information upon the network. Most key loggers concentrate upon stealing the passwords and gaining illicit access to the victim's bank accounts and credit card information. This kind of compromise of privacy is detrimental to the progress of web based trade and commerce, which would mean many users would be discouraged from trying out new e commerce applications with the fear of losing their money.

Key loggers are quite dangerous among all kinds of malware. They are not detectable by antivirus solutions. This is because they do not resemble viruses for all practical purposes.

Key loggers have a small footprint in terms of memory and processor utilization, making them practically untraceable for the users. Their code and construction is quite different from a normal virus or a worm.

Most key loggers are executable files written in C or C++. They are installed upon the system by administrator permission. They may not be detectable in the task manager. The log files that they write on to the system are hidden or undistinguishable from normal OS files[1]. To read the key logger log files, it is not even necessary for the adversary to have a physical access on to the system. They can simply have a spyware program to record and send the file to a remote system in a stealth mode [2].

II. A BRIEF DISCUSSION OF SOFTWARE KEYLOGGERS

Key stroke loggers are broadly classified into two types:

- Hardware key loggers
- Software key loggers

It is quite impossible to mitigate the effects of hardware key loggers from a software level. The only countermeasure for regular users is to avoid using their bank accounts from shared systems.

Software key loggers on the other hand can be tackled by some mechanisms, though quite difficult.

Software key loggers are categorized into several kinds:

1. *Hypervisor Based:* The malware is actually a hypervisor running under the operating system. Blue Pill is one such theoretical malware, which is supposed to be undetectable even when the algorithm of the malware is publicly known.[3]
2. *Kernel Based:* The malware process needs to run in either user space or kernel space. 95% of key loggers run in user space [4]. It requires a lot of expertise to develop a kernel based key logger. They are quite difficult to write as well as detect. [5]. They are implemented as root kits that gain access to the system resources and pose as a keyboard driver. This would require the malware process becoming a super user process making it difficult for normal processes with a lower level of access to detect them [5].
3. *API Based:* There are some vulnerabilities within the Windows API which would make it easier for adversaries to hook malicious programs. A windows hook is the core of a key logger. A *hook* is a point in the system message handling mechanism where an application can incorporate a procedure to have access to the messages reaching and leaving the windows process [1]. Windows APIs such as *GetAsyncKeyState( )* and *GetForegroundWindow( )* are utilized for this purpose.
4. *Packet Analyzers:* These malware capture the network traffic associated with HTTP POST events to read the passwords being sent in a plain text. HTTPS was introduced only to combat these kinds of attacks.

Several other kinds of software key loggers exist too such as Form grabbers and Memory injection based (Man in the Browser) attacks.

III. Existing Countermeasures And Related Work

The major threat of key loggers lies in the fact that they use keyboard input for authenticating the user. One possible solution would be to eliminate the usage of keyboard to enter passwords. This is possible by means of *graphical passwords* [6]. Graphical passwords involve the user clicking certain points on an image, sometimes by the help of a grid, in a certain sequence to produce the authentication scheme. The user needs to remember the points and sequence of clicking, which can be made easier by using a grid. It is obvious that the key loggers cannot act upon these authentication schemes.

Hook key loggers are eliminated to a certain extent by using some anti hook mechanisms as discussed in [1]. The method involves scanning all the running processes to see if there is a command that sets the hook. This involves the scanning of all dynamic link libraries related to the running processes.

Kernel level key loggers on the other hand are quite difficult to curb. Kernel level key loggers take different approaches. They may target the terminal to obtain the root password, which makes it easier for them to log in like any normal user. They may also have a mechanism to overtake normal kernel processes by posing themselves as legitimate kernel procedures. These threats are rare and grave, yet not entirely uncontrollable. [6] suggests several methods to bridge the gap at semantic level and creating a collaboration between the guest OS and the virtual machine.

Apart from these specific mechanisms, there are also the regular methods usually employed for malwares analogous to viruses. These include Signature based and Behavioral scanning techniques.

Signature based schemes focus upon the definitions of the malware structures. Definitions are checksum of the contents which are used to distinguish between each other. The working of a signature based scheme depends upon frequency of malware database updates. If a new virus with unknown definition attacks a system using signature based scheme, it may not be recognized as a malware before the update occurs. Hence it has all capability to attack the system and damage the resources as much as possible.

Behavioral schemes on the other hand focus upon malicious behavior of a particular software rather than its definition or signature. They constantly monitor the running processes to detect any suspicious activity and in that way. But it is not possible most of the times for these schemes to detect the behavior of key loggers since key loggers pose themselves as legitimate system processes and may not even have any obvious suspicious behavior. They may even rename themselves in order to avoid signature based scanning.

Hence, no technique of detection is fully potent against key loggers. We conclude that the best way to combat a key logger is only to avoid their effects by circumventing their attacks.

One such scheme is discussed in [8]. We have adapted the scheme for the mitigation of key logger attacks. We discuss the scheme below.

IV. Our Mitigation Mechanism

In order to mitigate the key logger attacks, we believe that the only solution is to mitigate their effects so that they become impotent to attack. For that purpose, we use the concept of a *Trusted Computing Base*. A trusted computing base is the collection of all the software , hardware and firmware components that are critical to the security of a system, in the sense that any vulnerability that occurs within the trusted computing base would affect the security of the entire system.

We employ a hypervisor as the trusted computing base for our project. A hypervisor, ideally, has to isolate its containing virtual machine operating systems. The advantage of such a hypervisor based mechanism is that the additional security layer can be easily added to the system without any modifications to the entire system architecture.

For this purpose, we employ a hypervisor, VMware, as is the case with our project and two virtual machines, each serving different purpose.

- *The trusted VM*, Linux operating system, Ubuntu in our case. The trusted VM is so called because, we use it minimally, making it very secure. It is free from any applications that may compromise its security. We use it only for the input of critical information.
- *The working VM*, Windows operating system, Windows XP in our case. The Working VM is used for all the day to day browsing activities. It may not be free from malware and it is not guaranteed to be secure.

The browsing activity happens in the working VM, and whenever the user needs to enter a password, he enters a dummy password instead. The dummy password is sensed by the trusted input activation module and the control is transferred to the Trusted VM, where the user enters the password without the fear of any malware and the concerned web page opens without any further distraction on the Working VM.

The entire process is shown in the following schematic diagram:

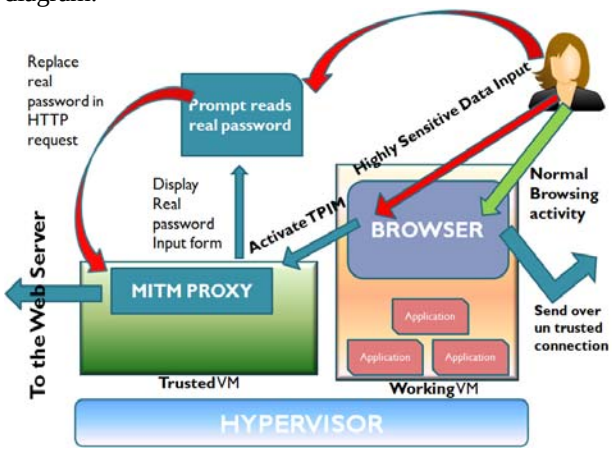


Fig 1: Working Diagram of TPIM Mechanism

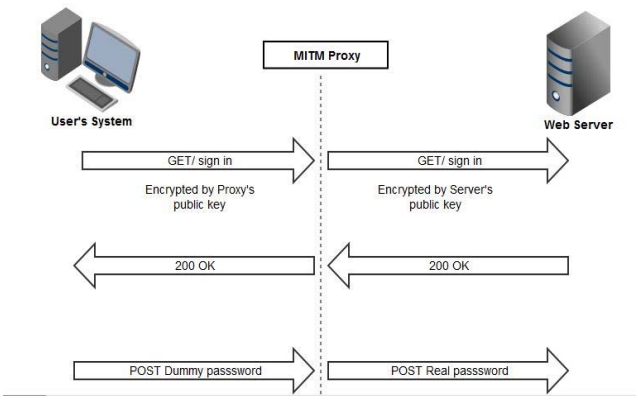
The reasons we have chosen Linux as a trusted VM rather that Windows are multifold:

- Kernel level key loggers that are the most dangerous kind are extremely difficult to develop in Linux.
- Windows Operating System is easier to attack in terms of System Hooks, whereas it is not possible in Linux systems.
- Newbie hackers mostly target the Windows systems rather than Linux systems [9].

There are several important components that make this model possible. The important one among those is the *Proxy Module,* which breaks the SSL/TLS connection of the browser and web server, with the permission of the user. Not in a malicious way, but a benign and helpful manner to mitigate the attacks of the system.

The proxy works as a man-in-the-middle application, acting as an interception between the two parties. It poses as the web browser to the web server and vice versa, and gains the public keys of both entities. In this manner, it gains to decrypt the passwords being sent by the browser. At that instant, it activates the Trusted Input module if it gathers a legitimate, pre- registered dummy password.

The working of the MITM proxy is shown in Fig. 2.



The steps of the process are as follows:

a. The proxy creates its own TLS connection to the service at the web server, at the exact IP address and destination port gathered from the CONNECT request.
b. It validates the certificate of the website and creates its own version with the "Common Name" from the service.
c. It signs the certificate with its own public key.
d. It then accepts the TLS connection from the user's browser environment.
e. It now has the ability to pass and forward the data back and forth between the browser and the web server.
f. By having the public keys of each of the entities, it is entitled to read, intercept and modify the data as it needs.

We can successfully implement this scheme using the special java APIs such as Java Cryptography Extension (JCE), Java Security packages and IAIK library implemented for java.

V. DISCUSSION

A. Advantages of the proposed scheme

a) We will be able to prevent a broad range of data stealing malware, not only key loggers but several other worms that target the banking applications.

b) Phishing Attacks can also be mitigated to a large extent. This is because, whenever a user is prompted by a phishing attacker to login to a fake login page, the trusted input proxy gets automatically activated and the data will be sent to the original page and not the fake page as prompted by the attacker..

B. Further points for discussion

We need to make sure always that the web browser makes use of the employed proxy. Without the use of the proxy, the entire scheme is defeated.

Also, other data stealing kind of malware like screen loggers (which record the screen activity) and mouse loggers, (which record the mouse activity) cannot be prevented.

Also, we work with an assumption that the hypervisor is completely secure and cannot be modified by any malware. This may not be practically true, as there may be attacks on hypervisor too, if not very common.

Finally, we have made use of Linux Operating system as our Trusted Domain, with the impression that it is entirely difficult to adversaries to create kernel level key logger attacks. It is difficult, but not impossible.

VI. CONCLUSION AND FUTURE WORK

We have discussed in a clear cut fashion the advantages and limitations of the system. But all through the limitations, we can guarantee the secure input of the password through this scheme. There is no need for any external hardware and it can work with the same computational cost as normal browsing activity. One time passwords, a regular counteraction scheme against key loggers needs the administrator of the system to be on a vigil always. But our scheme eliminates the need for administrator's intermediation, making it easier to be employed in domestic systems as well as business systems.

In our future work related to this project, we attempt to integrate the counteractive mechanisms for screen loggers as well as mouse loggers as well. Also, we propose to combat the kernel level key loggers in Linux to ensure the closure of all kinds of malware related to data stealing activity.

REFERENCES

[1] Muhammad Aslam, Rana Naveed Idrees, Mirza Muzammil Baig, and Muhammad Asif Arshad, "Anti-Hook Shield against the Software Key Loggers", National Conference on Emerging Technologies 2004.

[2] Muzammil M. Baig, W. Mahmood, A Robust Technique of Anti Key-Logging using Key-Logging Mechanism, 2007 Inaugural IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2007).

[3] http://theinvisiblethings.blogspot.in/2006/06/introducing-blue-pill.html

[4] Stefano Ortolani, Cristiano Giuffrida, and Bruno Crispo, "Unprivileged Black-Box Detection of User-Space Keyloggers" IEEE Transactions on Dependable and Secure Computing, Vol 10, Jan-Feb 2013.

[5] http://en.wikipedia.org/wiki/Keystroke_logging

[6] M. N. Doja and Naveen Kumar, Image Authentication schemes against key logger spyware,9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing

[7] Jesus Navarro Enrique Naudon Daniela Oliveira, Bridging the Semantic Gap to Mitigate Kernel-level Keyloggers.

[8] Manabu Hirano, Tomohiro Umeda, Takeshi Okuda, Eiji Kawai, Suguru Yamaguchi, T-PIM: Trusted Password Input Method against Data Stealing Malware, 2009 Sixth International Conference on Information Technology: New Generations.

[9] "What. Are key loggers? The essential 101" by Aditya Lad, published upon www.searchsecurity.techtarget.in.