ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st & 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

A New Method for Protecting User Mode from Root Kit Malwares

K.Rajan[1], D.Raghu Raman[2]

Department of Computer Science, Arunai Engineering College, Tiruvannamalai, Tamilnadu, India[1]

Department of Computer Science, Arunai Engineering College, Tiruvannamalai, Tamilnadu, India[2]

ABSTRACT- The dominant operating system in the world today is windows. There are some of the weaknesses present in the window architecture. Using this weakness root kit malware wants to utilize an administrative control of the windows, root kit malwares refers to software that is used to conceal the presence and permit an attacker to take control of a system. So, an attacker can capture the sensitive information that present in a system. To reduce the number of root kit injection first, we classify the legitimate and suspicious code using an algorithm if the process is a legitimate one means that the legitimate process is directly permitted to get the system service through the ntdll.dll which acts as a gateway to the kernel mode from the user mode. If it is a suspicious code means, it will be processed through the customized ntdll.dll. Monitor program is used to customize the ntdll.dll by hook.dll, using which the pre-validation and validation function is added in the ntdll.dll. Pre-validation is done by generating password for a suspicious code using a scrambling technique, then by using we unscramble the dispatch-ID which was scrambled in the user mode and redirect the control to the validation function if it matches with any of the system services, otherwise the control will be disallowed. It provides an additional protection that avoids the system crash and allows only the legitimate program to accomplish the system services.

KEYWORDS: root kit malware, hook, dispatched

I. INTRODUCTION

A root kit is stealth software which is designed to hide the existence of certain processes or programs from normal methods of detection and enable continued privileged access to a computer. Root kit installation is either automated, or an attacker can install this malware once they've obtained root or Admin access. Obtaining this access is a result of direct attack on a system. Once installed, it becomes possible to hide the intrusion as well as to maintain privileged access. The key is the root/Admin access. Full control over a system means that existing software can be changed, including software that might otherwise be used to detect or circumvent it.

Root kit detection is much difficult because a root kit may be able to subvert the software that is intended to find it. Detection method using an alternative and trusted

operating system, behavioral-based method, signature scan, difference scanning, and memory dump analysis method. Removal can be difficult or practically impossible, particularly in cases where the root kit resides in the kernel; reinstallation of the operating system may be the only available solution to this problem. When dealing with firmware root kit, removal may require hardware replacement, or specialized equipment or a tool.
In computer programming, the hooking covers a range of techniques used to alter or augment the behavior of an operating system, of application, or of other software parts by intercepting function calls or messages or events passed between software components. Code handles such intercepted function calls, events or messages is called a "hook".

Hooking technique is used for lot of purposes, including debugging and extending functionalities. Example include intercepting keyboard or mouse event messages before they reach an application, or intercept theoperating system calls in order to monitor behavior or modify the function of an application or other component. Hooking can also be used by malicious code or by malwares. For example, root kits, pieces of software that try to make themselves invisible by faking the output of API calls that would otherwise reveal their presence, often use hooking technique.

API hooking based method can be divided into two kind: user-mode APIs hooking and another is kernel-mode APIs hooking. User-mode APIs hooking method forbid injected code calling user- mode APIs. While these APIs are called, the corresponding mechanism will be watched. The API call will be then reviewed and justified. Even though, if injected code doesn't call any user-mode APIs, this method can't stop illegal system requests. Kernel-mode APIs hooking method solve this loophole at some point of system services request happen in kernel mode, the corresponding mechanism will inspect the requests and decide whether to allow the requests.

II. BACKGROUND AND RELATED WORK

A virtual machine (VM) can be simply created upon use and disposed upon the completion of the tasks or the detection of error. The demerits of this approach is that if there is no malicious activity that the user has to redo all of the work in her actual workspace since there is no easy way to commit (i.e., merge) only the benign updates within the VM back to the host environment. A practical application is to allow user to install and try new application without worrying about malware. In other words, if abnormal happens, one can easily throw away the infected VM. One disadvantage of this approach is that if all processes run normally within a VM and there is no malicious activity, a user has to redo all the work in actual workspace since, there is no secure commitment mechanism to save the benign changes within the VM back to the host environment

Zhiyong Shan et al.[1] designed secom which have five approach to overcome the above problem, it propose the first secure commitment approach, Secom, for Operating Systen-level VM to identify compromised OS objects and selectively merge only legitimate changes into the host. Moreover, it three novel features allows it to complete the task in a lightweight but efficient manner.

It proposes a novel clustering approach to segregate benign and malicious changes within a VM. The approach relies on starting and tracing rules to trace OS-level flows to collect modifications, and the labeling method to group collected changes into cluster.It gives a new behavior-based malware detection approach. A suspicious clusters are considered to be malicious only when it exhibit atleast two types of malware behaviors. Moreover, as all cluster in Secom are derived from hazardious sources, our proposed detection procedure implicitly takes into an account that the source of all processes that launch the behaviors. This experiment showed that the number of false positives (FPs) of this method is much smaller than that of existing online malware detection approaches. We have implemented a prototype of Secom on a feather-weight virtual machine (FVM) on Windows. Experiment show that it can effectively filter out a number of real-world malwares while imposing only a small overhead on the FVM. Moreover, it filters malware objects more thoroughly than that of commercial antivirus software.

To increase the survivability and concealment of malware, malware writers have developed various approaches to fight against distinct security solutions. Malware that terminates the execution of antivirus software without the consciousness of the antivirus software users is called an antivirus terminator. Without the protection of terminated security tools, an attacker can do anything on the intruded host. Antivirus terminator used seven method such as Process termination method, Null debugger method, Dll unloading method, Close message method, Mouse simulator method, Registry modification method and Thread termination method. Hence, developing a solution to protect antivirus software against antivirus terminators becomes a critical issue.

Fu-Hau Hsu et al.[2] implemented a approach called ANtivirus Software Shield (ANSS), to protect antivirus software against antivirus terminators. ANSS uses SSDT hooking to intercept specific Windows APIs and analyzes their parameters to filter out hazardous API calls that will terminate antivirus. ANSS intercepts frangible API call before their execution so that ANSS can monitor frangible API call. When a program issues a system call to execute a system service, kiSystemService look up on the SSDT to find the address of the corresponding system service. Then kiSystemService uses the address to invoke the system service. Through SSDT hooking, ANSS modifies some function addresses stored in the KiServiceTable service descriptor table and replaces them with ANSS API handlers. After an ANSS API complete its task, it invokes the original API.

Interceptor: It intercepts the execution flow of the program and transfers the execution flow to the filter before the code of a frangible API is executed.

Filter: An invocation to a frangible API is transferred to its corresponding ANSS API first through SSDT hooking. Each ANSS API enforces the rule applying to its related frangible API.

Drive-by-Download attacks are one of the most severe security threats to computer and network system. A user using a vulnerable browser or browser plug-in may become a victim of a drive-by-download attack, thus an attacker can download and execute any code on the victim's host. A drive-by-download attack is launched through a web page with crafted malicious contents. The web server that host the web page may be owned by an attacker or may be compromised by an attacker or may be a normal benign host which allows other persons to put their contents, such as an advertisements, in the web page of the host. To accomplish drive-by-download attack, Malware Bootstrap Function (MBF) must be injected into the address space of the attacked browser. Then the execution flow must be transferred to the MBF through some vulnerability in the browser or a plug-in in the browser. In turn, the MBF will download malwares into the compromised host and execute the malware.

Chang-Kuo Tso et al.[6] proposed a scheme called Browser Guard a runtime, behavior based solution to drive by download attack. Browser Guard analyzes the download scenario of every downloaded object. Based on the download scenarios, Browser Guard blocks executions of any executable file that is downloaded to the host machine without the consent of a user.

Browser Guard consists of a Browser Guard-BHO in every IE process, a Browser Guard-Kernel in the kernel space, and a list server process. The list server process contains two lists, a white-list and a blacklist. The white-list records the URLs of benign files and the hash vales of benign executable files. The blacklist records the hash values of detected malicious files.

III. SYSTEM DESIGN

A. ALGORITHM:

An algorithm used to classify the legitimate and suspicious code

Take active processes as a input. if the process is a legitimate one means that the legitimate process is directly permitted to get the system service through the ntdll.dll

which acts as a gateway to the kernel mode from the user mode. This classification used for a legitimate user get the service from kernel mode directly through the ntdll.dll in the user mode. If it is a suspicious code means, it will be processed through the customized ntdll.dll

SYSTEM ARCHITECTURE

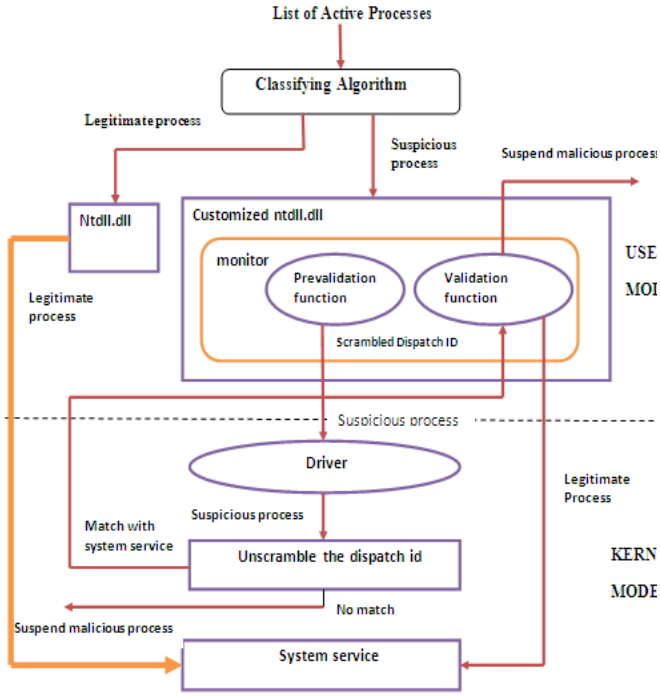

Fig 1

B. MONITOR:

After the process is loaded and initialized, the monitor suddenly suspends the main thread of the protected process. The monitor includes the PID of the process into the monitor's list. Monitor program is used to customize the ntdll.dll by hook.dll, using which the pre-validation and validation function is added in the ntdll.dll. We replace the sysenter commands by the commands jmp pre-validation. The prevalidation function in the system has three missions they are caching registers, preparing the password and saving stack pointer. Pre-validation is done by generating password for a suspicious code using a scrambling technique; the dispatch id of a native API function is scrambled at this pre-validation function. The scrambled dispatch id and forwarded to the kernel mode.

(When a kernel mode receives the request from the process, the driver will authenticate the process with the help of validation function present in the user mode to protect the malicious code that directly attack the kernel mode.)

C. DRIVER:

Before handling a request, the driver has to determine if the request has comes from legitimate code or from suspicious code. If the request comes from legitimate code (i.e from ntdll.dll), it will get the system service directly. If the request comes from suspicious code (i.e from customized ntdll.dll), then the driver has to determine if the request has been authenticated by the validation function yet.

If this request has not been authenticated. the driver records the dispatch ID, its process ID, and thread ID. Then unscramble the dispatch-ID which was scrambled in the user mode. Redirect the control to the validation function if it matches with any of the system services by changing the return address to where validation function located. This can be done by modifying values of a fixed position in the stack of the kernel mode, from that original value esp to the new value R.When the driver carries out a ret instruction, it exit the kernel mode and restores eip to the address of the validation function. The function then return to the user mode and executes the validation function. Notice that esp will be pointed to an empty address R _ 1.If the process has been authenticated by the validation function already. The driver is able to retrieve this record by matching the service number, the process ID, and the thread ID. In this case, these records are removed and the API request is served.

D. VALIDATION FUNCTION:

Pop out the scrambled dispatch-ID in pre-validation functions and unscrambles it to compare with the unscrambled dispatch-ID comes from driver. If matches, then it is legitimate and get the system service, otherwise the request may come from malicious. So it will be suspended.

V. CONCLUSION

This method uses an algorithm to classify the legitimate and suspicious process then it allows the legitimate process directly to gain the system service from the kernel mode and the suspicious code allowed getting the service through customized ntdll.dll. That the customized dll used to scramble the dispatch id. In this paper, we proposed an algorithm and a method which create a password using prevalidation function which protects the user mode functions through which the kernel mode service are provided. We believe this method is a better method than the other existing methods in protecting API functions of the windows. These methods effectively prevent the native API to give the system service to the unauthorized user. Thus in the kernel mode it effectively prevent malicious code to access the native APIs. This method does not provide the solution if the malicious code affects the user mode in the operating system. Thus the malicious code can access the lower level kernel API by using hooking techniques. we proposed a scheme in kernel mode to protect Windows system against malicious code. This scheme prevents malicious code from directly accessing kernel APIs. We believe this scheme is currently the best real-time solution for Windows system in this layer.

REFERENCES

[1] Zhiyong Shan, Xin Wang, Tzi-cker Chiueh, "Malware Clearance for Secure Commitment of OS-Level Virtual Machines," IEEE Transactions on Dependable and Secure Computing, vol. 10, no. 2, pp. 70-83, March-April 2013.

[2] Fu-Hau Hsu, Min-Hao Wu, Chang-Kuo Tso, and Chieh-Wen Chen. "Antivirus Software Shield Against Antivirus Terminators", IEEE transaction on information forensics and security, vol. 7, n0. 5, October 2012

[3] Jinku Li, Zhi Wang, Tyler Bletsch, Deepa Srinivasan, Michael Grace, and Xuxian Jiang. "Comprehensive and Efficient Protection Of Kernel Control Data" IEEE transactions on information forensics and security, vol. 6, pp.1404-1417, december 2011

[4] *Trent Jaeger, Paul C. van Oorschot, Glenn Wurster. "Countering unauthorized code execution on commodity kernels: A survey of common interfaces allowing kernel code modification" ELSEVIER Articles on Computers and Security Volume 30,_Pages 571–579, Issue 8, November 2011*

[5] *Arati Baliga, Vinod Ganapathy, and Liviu Iftode." Detecting kernel-Level Rootkits Using Data Structure Invariants" IEEE transactions on dependable and secure computing, vol. 8, pp. 670-684, september/october 2011*

[6] *Fu-Hau Hsu, Chang-Kuo Tso, Yi-Chun Yeh, Wei-Jen Wang, and Li-Han Chen. "BrowserGuard: A Behavior-Based Solution to Drive-by-Download Attacks" IEEE journal on selected areas in communications, vol. 29, pp.1461-1468, august 2011*

[7] *Hung-Min sun, Hsun Wang, King-Hang Wang and Chien. "A Native APIs Protection Mechanism in the Kernel Mode against Malicious Code", IEEE transaction on computers, vol. 60, no. 6, June 2011*

[8] *Desmond Lobo, Paul Watters, Xin-Wen and Wu and Li Sun, "Windows Rootkits: Attacks and Countermeasures" IEEE Proceedings. Second Cybercrime and Trustworthy Computing Workshop (CTC 2010), july 2010*

[9] L. Nguyen, T. Demir, J. Rowe, F. Hsu, and K. Levitt, "A Framework for Diversifying Windows Native APIs to Tolerate Code Injection Attacks," Proc. Second ACM Symp. Information, Computer and Comm. Security (ASIACCS '07), *pp. 392-394, 2007.*

[10] LI Xianghe, ZHANG Liancheng, LI Shuo "Kernel Rootkits Implement and Detection" Wuhan University Journal of Natural Sciences Vol. 11, pp.1473-04, 2006

[11] *Mohan Rajagopalan, Matti A. Hiltunen, Trevor Jim, and Richard D. Schlichting. "System Call Monitoring Using Authenticated System Calls." IEEE transactions on dependable and secure computing, vol. 3, pp.216-229, july-september 2006.*

[12] *John G Levine, Julian B.Grizzard and Henry L.Owen, "Detecting and Categorizing Kernel-Level Rootkits to Aid Future Detection" IEEE Security & Privacy, february 2006*