International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Detecting the Rootkit through Dynamic Analysis

D. Suganya Gandhi[1], S. Suresh Kumar[2]

[1, 2]Rajalakshmi Engineering College, Anna University, Chennai, India

Abstract: ***Network security provides a security for all the programs or files or system. Some attackers attack a programs or files or passwords or other personal details of the user. Like the same way Rootkit is one of the malicious file or a software which attacks a network security and acts an administrator in an absence of the user knowledge. Rootkit virus is stealthy in nature and is installed in the system through a file or a driver or coding. It attacks the system through the kernel-level in the real time. Files are hided through the rootkit in the absence of the user knowledge. They can monitor the other user's activity when the botnet is installed in the other system. Rootkit allows the attacker through the backdoor. So that attacker can steal the users personal details. Task manager, service and the registry are got destroyed or made changes. The attacker can make any changes at any time. Finally the malicious file and authorized files are distinguished and their accuracy is performed.***

Keywords: Malicious, Rootkit, Static analysis, Kernel-level

1. Introduction

Nowadays many malwares are very harmful so that they can't be easily detected and prevented. It becomes the serious issue in the growth of the technology. The various modules regrets with newer malwares based on that one of the important modules is rootkit module. Over the observation overall current malware contain 10% of rootkit may present a small percentage of the total malware population. Among that the rootkit is most dangerous malware. Generally rootkit is developed as itself to hide it from rest of the modules in the system. The malware industry has three major effects such as hiding technology, less concerned with their size on the victim's hard disk and storage space on disk. It can occur in the windows, Linux etc. It can affect all parts of the system. The rootkit in the industry are botnet management, through this the attacker can monitor or view all the programs or files running on the system in the absence of the user and can act as an administrator. It has the same privilege as the operating system. [1].The rootkit technology in legitimate applications that trend to employ security products, copy protection technologies and recovery tools [2].Rootkits are discussed based on the operation system and employs API's to use system services to communicate with the operating system.

Rootkit has two levels: User level and kernel level. User-level rootkits are programs that overwrite the file system binaries and libraries with customized versions that accomplish the desired "hiding" goals. User level has the ability to control and assign what users can and cannot do. It must manage and allow access to such functions as writing and editing posts, creating pages, defining links, creating categories, managing plug-in, managing themes and managing other users. It executes user-space code. It can call into kernel code at elevated security levels. It runs on the top of the system in user-mode [3]. Many network services these days now run as restricted user-level process. This means when a remote hacker breaks into such a service, they do not get full control over the machine. They might be able to define a web page or cause other havoc, but they do not own the box. At this point, the intruder will need to run some sort of privilege escalation exploit in order to root the system.

Kernel rootkits are a special category of malware that are deployed directly in the kernel and hence have unmitigated reign over the functionalities of the kernel itself. Next the process use of HFS (Hidden File System) [4] and the other firmware modification methods for a good review of rootkits [5]. It runs on the kernel code and is not associated with a user-space process. Kernel level runs multiple processes at a time. It connects the application software to the hardware of a computer. A rootkit can modify your software programs for the purpose of infecting it with spyware. The spyware that is installed by the rootkit is sometimes difficult to detect however, you will notice strange things happening like links appearing on desktop and changes in the habits of your web browser. A back door is a modification that is built into a software program in your computer that is not part of the original design of the program. It creates a hidden feature in the software program that acts like a signature so the intruder can use the software for malicious purposes without being detected. Bytes are constructed in a specific order which can be modified by a rootkit. If the bytes are rearranged it compromises the computer software protections so the intruder can gain control of the software for malicious purposes. Source code modification is accomplished by modified the code in your PC's software right at the main source. The intruder inserts malicious lines of source code for the purpose of hacking software with confidential information. The code can also end up in a myriad of other programs which makes it very difficult to locate.

The most basic include searching for modified kernel modules on disk, searching for known strings in existing binaries, or by searching for configuration files associated with specific rootkits. The problem is that when a system has been compromised at the kernel level, there is no guarantee that these tools will return reliable results. Apart from that, they are several problems also identified and discussed below:

Symbolic execution is a static analysis technique in which program execution is simulated using symbols, such as variable names, rather than actual values for input data. The program state and outputs are then expressed as mathematical (or logical) expressions involving these symbols. When

performing symbolic execution, the program is basically executed with all possible input values simultaneously, thus allowing one to make statements about the program behavior.

One problem with symbolic execution is the fact that it is, due to the halting problem, impossible to make statements about arbitrary programs in general. However, it is often possible to obtain useful results in practice when the completeness requirement is relaxed. Relaxing the completeness requirement implies that the analysis is not guaranteed to detect malicious instructions sequences in all cases.

Control flow instructions present problems for our analysis when they have two possible successor instructions (i.e., continuations). In this case, the symbolic execution process must either select a continuation to continue at, or a mechanism must be introduced to save the current machine state at the control flow instruction and explore both paths one after the other. In this case, the execution first continues with one path until it terminates and then backs up to the saved machine state and continues with the other alternative.

One problem is caused by the exponential explosion of possible paths that need to be followed. Consider the case of multiple branch instructions that are the result of a series of if-else constructs in the corresponding source code. After that, each if-else block, the control flow joins.

2. Related Work

Over the recent years various rootkits detection techniques have been proposed from the time of first known rootkits. The single defining characteristics of a rootkit are stealth. Rootkit can be defined as the set of tools or programs which patch and Trojan existing execution paths within the system. We have discussed about various virtualization techniques on the end user system based on some limitations [6].

Kernel Integrity checks and cross view is the two dynamic detection approaches which are unable to detect a wide range of rootkits. The previous work focused on malicious rootkit driver detection which can be categorized, based on the approach used to define the distinguishing features, as either blind or behavior driven.

Schmidt et.al [7] deals with the gather function calls observed in the code and select a small set as distinctive features based on statistical analysis. Rootkit hooking techniques often focus on passively hiding processes or files, filter drivers are usually engaged in actively intercepting user data. Common uses for filter drivers are logging keystrokes or network activity to capture user passwords or other sensitive information. When a victim visits this page, the script is executed and attempts to compromise the browser or one of its plug-in. To detect drive-by-download exploits, researchers have developed a number of systems that analyze web pages for the presence of malicious code. Most of these systems use dynamic analysis.

That is, they run the scripts associated with a web page either directly in a real browser (running in a virtualized environment) or in an emulated browser, and they monitor the scripts' executions for malicious activity. While the tools are quite precise, the analysis process is costly, often requiring in the order of tens of seconds for a single page. Therefore, performing this analysis on a large set of web pages containing hundreds of millions of samples can be prohibitive. One approach to reduce the resources required for performing large-scale analysis of malicious web pages is to develop a fast and reliable filter that can quickly discard pages that are benign, forwarding to the costly analysis tools only the pages that are likely to contain malicious code.

According to Sami at al. use to find the frequent API call sets that has been the features on such approaches result in a large number of features which results in reducing the classification efficiency. On the other hand the existing system have employed the fisher score, random projection, information gain, and feature-hashing respectively, in order to decrease the large number of features obtained and select the more important features[8].

In Prophiler work is one which has 77 features are proposed for finding the malicious behaviors in WebPages. Similar to it Zhao et al. Extract a FCG (i.e. function call graph) from a file for a malware behavior as opposed to a statistical feature selection process. They also discussed about the semantic-aware detection and on the detection of kernel-level rootkit drivers by modeling improper kernel memory accesses. Malware classifiers often use sparse binary features, and the number of potential features can be on the order of tens or hundreds of millions. Feature selection reduces the number of features to a manageable number for training simpler algorithms such as logistic regression, but this number is still too large for more complex algorithms such as neural networks. To overcome this problem, the project adopts random projections to further reduce the dimensionality of the original input space. Using this architecture, it is possible to train several very large-scale neural network systems with over 2.6 million labeled samples thereby achieving classification results with a two-class error rate of 0.49% for a single neural network and 0.42% for an ensemble of neural networks.

Once the rootkit is publicly known, Anti-virus software can develop a signature for it. It changes to the operating system may be detectable using memory scans that look for changes to critical operating system components in memory. Therefore the advantages of the rootkit to be able to hide the file or document and make changes over the operating system [8].

Anomaly detection [9] has also been applied to rootkit detection in various forms. It defines the normal system characteristics or behavior. This detection may be used to examine the structural characteristics of functions to detect hooking. Table based hooking methods are also detected.

The concept of manipulating the windows kernel for stealth started with NTRootkit-A which hooked SSDT. It extends the idea by hooking the dispatch table. Then it became

extremely popular among rootkits and is still being used. The rootkit authors soon realized that modifying an existing kernel driver would benefit them further in stealth. Rootkit quickly took advantage of this technique by overwriting less critical drivers like beep system [10].

3. Proposed Scheme

Our proposed system is based on the modern kernel-level rootkit behaviors by tracing it activity in the file level systems in the kernel. By this approach we can able to identify the sign of rootkit activity by modifying the memory access and file system. Because there has been no prior work which employs static analysis to extract pre-defined behavioral features covering different kinds of kernel-level rootkit drivers.

The proposed system deals with installing the rootkit and affects the system and affects the other user's activity. Our system took Windows kernel-level rootkits by differentiating their malicious behavior to that of the legitimate file drivers. Our proposed system is evolved by the continuation of earlier work such as low cost static analysis is inefficient as well as the malware detection techniques trust the operating system for run-time analysis by default. These systems process the works of Static Detection of Rootkit by the behavior of trends & File Activity. These techniques can be implemented in kernel space, but with more constraints and limitations on facilities it can't be processed. Which also affects the memory space which is shared by many other kernel modules and the operating system itself? The above analysis, result in simple light weight static detection technique which was monitor by the by a user-level application. As a final result it classifies the file system in windows kernel as either malicious or legitimate.

The following points explain about the advantage of proposed system:

- Extracting the predefined features of Kernel level drivers and making them legitimate
- Windows kernel-level rootkits according to current day trends
- Behavioral differentials between the malicious & legitimate driver
- Memory tracking
- File modification & effectively analyzing the malicious kernel driver
- It use headers that test terminal is permitted to send.
- Reduce the downtime and work in real networks & < 1% of link overhead

4. Architectural Design

Kernel level connects the application software to the hardware of a computer. It runs multiple processes, whereas the user level executes the kernel code at the elevated security levels. It runs kernel code and is not associated with a user-space process. In proposed system, the rootkit is injected and detected in the kernel level. So the kernel level can be attacked by the rootkit through the device drivers, device I/O, network protocols, file system and debugging facilities. But it is very difficult to attack and detect the kernel. Rootkit is a type of software that is designed to hide the existence of certain processor programs. It gives the negative idea to destroy the computer.

The system architecture explains that the kernel level can be attacked by the file system. All the files are stored in the database which may be pdf, document, text or exe etc. Files are used to hide in kernel level so that the rootkit can modify some files without the user knowledge.

If a rootkit has replaced the part of the kernel servicing those calls, it can return all the information the system monitor wants – except for anything relating to the rootkit. Rootkit are installed through the code to detect the kernel. It may affect the task manager which shows all the running programs in the system, registry which has all the registered or recorded files and also many services in the system. Rootkit files are mixed with the authorized files to hide from the user. It hides many processes running on the system.

Once the rootkit affect the files in the task manager, registry and services, it may detect through the kernel level. It is detected by using the rootkit removal tool so that it list out the modified files.
Antivirus programs are very poor at detecting rootkits on a running system. This ability to operate invisibly within the operating system means that a major use of rootkits is to conceal other malware. Some rootkits may disable antivirus software. The best way of detecting the rootkit in the operating system is to shut down the operating system itself and examine the disk upon which it is installed.

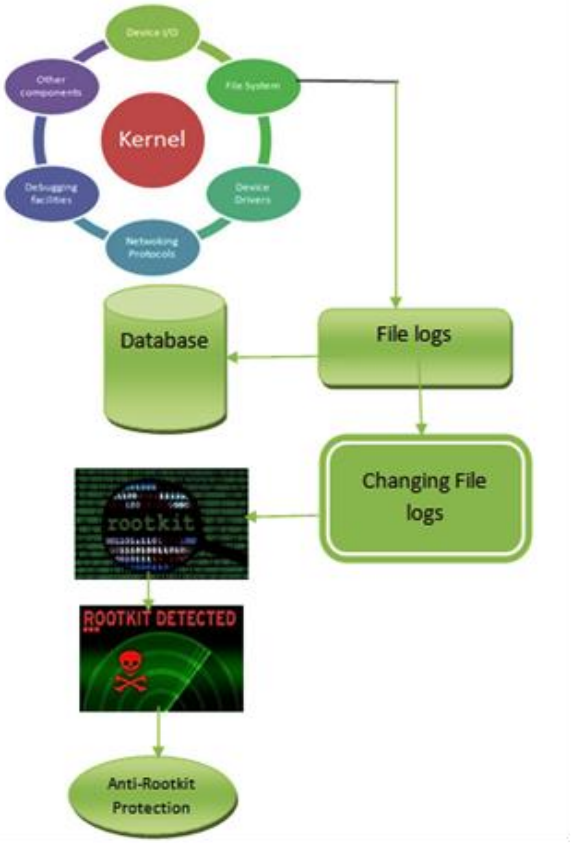

Figure 1: System Architecture

5. Methodology

The implementation section has several phases on the initial stage is on OS model. The window kernel system is analyzed and the total resources are traced and stored in the database. The overall resource availability is monitored at the initial stage. We first we observe a range of trends in the functionalities provided by rootkit. The functions included in that are injection, file system behavior, kernel memory over-writing and file system modifying. In which a malicious may have one or more of the above noted functionalities implemented in the system. Some of the noted functionalities are;

1. Injection
2. File Activity
3. Installing Rootkit on Windows
4. Rootkit on the Oracle VM
5. File Monitoring

4.2.1 Injection

Injection is a kind of patching a repair module in the kernel level as the most popular behaviors in rootkits. There are various methods to inject into user-level processes one is inject a desired code via allocation of Virtual Memory another is creating and mapping of a new section into a process memory space. By tracing the memory behavior the anomality of the hidden rootkits is captured.

4.2.2 File Activity

As stated on the earlier stage the windows kernel system and the overall resource availability is stored in the database. The overall behavior is monitored if the hidden rootkit had change the behavior of the file activity in the system than it was traced and the anomalistic behavior is rectified to achieve a free from malware OS model by our proposed system. It can be achieve by Windows does include some function calls for accessing files for situations like when a driver is going to update hardware or software that should be handled carefully. So the rootkit may indulge to handle or modify a file including logs and/or spying data, or alter file access times so such behavior should be traced and observed on the system.

4.2.3 Installing Rootkit on Windows

Once the control panels are installed on the system, it starts the front end and back end of the panel. While starting the control panel, it switches on the port number and port id for the easy reference. Create the database in the name of the bot. When the database is created it starts the execution in the file or folders or in the program. Once the rootkit is installed in the name of the bot it may affect any of the files or may affect the system

4.2.4 Rootkit on the Oracle VM

Like the same of the rootkit installed in the windows, also the oracle vm can also install the rootkit as the same way. The difference between the oracle VM and windows is the rootkit alone installed in windows whereas the oracle installs the rootkit with the anti-virus.

4.2.5 File Monitoring

After installing the rootkit, the attacker monitors the other user activity through the botnet. Attacker monitor others activity in the absence of the user. Rootkit itself act as an administrator.

5. Discussion

The rootkits are various types our proposed system effectively detect malicious rootkit drivers accurately but approach will not be effective for all the rootkits. In the kernel space without deploying any driver un-patched vulnerabilities could be employed by the rootkit. The rootkits in the hidden file systems which can be load them into memory on each boot. There is any kind of penetration into kernel space should done or not must be noted. The kernel vulnerabilities, requires at least an initial kernel driver to be loaded based on that we can able to detect the loader kernel driver at the beginning of the process. The proposed static analysis technique needs number of assumptions which makes anything loaded on the system otherwise it protected by Kernel-level self-protection solutions. The changes in the feature category show the rootkit attack that happened on file system or not.

6. Conclusion

On this static analysis for detecting kernel-level rootkits two observations are made such that windows kernel level space and another one is kernel level code. Among the observation the proposed system enables the rootkit detection in malicious drivers. In our static analysis approach the main advantage is as in dynamic analysis, is that it does not require the binary being analyzed to be executed. On it's the general behavior of the rootkit shows the level of suspicious activity present, such as hiding intent. Which traces effectively if any behavior happened in the file system by the rootkit would examined by its various features. In future this work must be continued on dealing with various driver modules on the operating systems.

7. Acknowledgement

Our completion of this paper could not have been accomplished without the support of staff members those who are working with us. We are very much thankful to them. For the reference, we refer many articles and research papers of many authors and institutions those are available in online and offline. We offer our sincere appreciation for the learning opportunities provided by those authors and institutions. At last but not least, our heartfelt thanks to all our family members for caring us and for the huge support.

References

[1] A. Kapoor and R. Mathur, "Predicting the future of stealth attacks," in Virus Bulletin conference, 2011.
[2] (2012) Zegost - analysis of the Chinese backdoor. [Online].Available: http://artemonsecurity.blogspot.com/2012/12/ zegost-analysis-of-chinese-backdoor.html

[3] P. Gutmann, "The commercial malware industry," in DEFCON conference, 2007.
[4] F. Op. (2008) the fu rootkit. [Online]. Available: http://www.hackerzvoice.net/ceh/CEHv6%20Module% 2007% 20System%20Hacking/FU Rootkit/
[5] F.-S. Lab. (2005) Cut'n'paste rootkit-bots. [Online]. Available:http://www.fsecure.com/weblog/archives/00000559.html
[6] S. S. Response. (2005) W32.mytob.ar. [Online]. Available: http://www.symantec.com/security response/writeup.jsp?docid=2005-041116-0718-99
[7] K. Kasslin, M. St°ahlberg, S. Larvala, and A. Tikkanen, "Hiden seek revisited–full stealth is back," in Proceedings of the 15th Virus Bulletin International Conference, 2005.
[8] J. S. Center. Ghost security suite ssdt hooks multiple local vulnerabilities. [Online]. Available: http://www.juniper.net/security/auto/vulnerabilities/vuln25709.html
[9] Kaspersky internet security 6 ssdt hooks multiple local vulnerabilities. [Online]. Available: http://www.juniper.net/security/auto/vulnerabilities/vuln24491.html
[10] M. Russinovich. (2011) Using rootkits to defeat digital rights management. [Online]. Available: http://blogs.technet.com/b/markrussinovich/archive/2006/02/06/using-rootkits-to-defeat-digital-rights-management.aspx
[11] (2011) Returnil ssdt hooks listed as ¡unknown¿. [Online].Available:http://www.wilderssecurity.com/showthread.php?t=303964
[12] V. Rusakov. (2011) Legit bootkit. [Online]. Available:https://www.securelist.com/en/analysis/204792203/Legit bootkits
[13] E. Rodionov. (2012) Win32/gapz: New bootkit technique. [Online]. Available: http://www.welivesecurity.com/2012/12/27/win32gapz-new-bootkit-technique/
[14] E. Rodionov and A. Matrosov, "Defeating anti-forensics in contemporary complex threats."
[15] S. Embleton, S. Sparks, and C. C. Zou, "Smm rootkit: a new breed of os independent malware," Security and Communication Networks, 2010.
[16] D. Zovi, "Hardware virtualization-based rootkits," Black Hat USA, 2006.
[17] S. Sparks, S. Embleton, and C. Zou, "Windows rootkits a game of hide and seek," Handbook of Security and Networks, p. 345, 2011.
[18] C. Kruegel, W. Robertson, and G. Vigna, "Detecting kernel-level rootkits through binary analysis," in Computer Security Applications Conference, 2004. 20th Annual. IEEE, 2004, pp. 91–100.
[19] McAfee. (2011) Root out rootkits, an inside look at mcafee deep defender. [Online]. Available: hhttp://www.intel.ph/content/www/xa/en/enterprise-security/mcafee-deep-defender-deepsafe-rootkit-protection-paper.html
[20] S. Grobman et al., "Method and apparatus to detect kernel mode rootkit events through virtualization traps," Nov. 30 2010, us Patent 7,845,009.

Author Profile

Suganya Gandhi.D is currently a PG scholar in Computer Science and Engineering from the Department of Computer Science at Rajalakshmi Engineering College, Chennai. She received his Bachelor Degree in Computer Science and Engineering from S.A Engineering College, Chennai and Tamilnadu. Her Research areas include Wireless Networks and Distributed Computing.