IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2320-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 10, Issue 5, May 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.512

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 5, May 2021 ||

DOI:10.15680/IJIRSET.2021.1005014

A Survey on Rootkit Techniques

Aniket Sahu[1], Animesh Tarodia[2], Sagar Jagtap[3], Miss Rajeshwari Gundla[4]

Student, School of Computer Science and Engineering, Ajeenkya D Y Patil University, Pune, India[1]

Student, School of Computer Science and Engineering, Ajeenkya D Y Patil University, Pune, India[2]

Student, School of Computer Science and Engineering, Ajeenkya D Y Patil University, Pune, India[3]

Assistant Professor, School of Engineering, Ajeenkya D Y Patil University, Pune, India[4]

ABSTRACT: Rootkit is one of the most critical problems in network communication systems, as it relates to Internet users' protection and privacy. Since the operating system has a back door, a hacker may use rootkit to assault and invade other people's computers, enabling him to easily capture passwords and message traffic to and from these computers. With the advancement of rootkit technology, its implementations are becoming more and more diverse, making detection more difficult. Furthermore, rootkit detection technology knowledge and effective tools are still relatively scarce for a variety of reasons, including trade secrets, difficulty in growth, and so on. We analyze and survey rootkit techniques at both the user and kernel levels in this paper.

KEYWORDS:Rootkit, Hooking, API, Kernel

I. INTRODUCTION

The superlative invention of the web within the twentieth century has generated innovative developments in the computer world, whereas web speed is increasing, adverse effects are also increasing. A malicious program, which might have taken an extended time to propagate in former days, can now propagate in a few seconds or minutes. In the past, virus spreading technology was easy and the route was short, giving an account technician enough time to defend themselves from the virus. To deal with their security risks, security experts compile, disassemble, and evaluate malicious code. They can track and analyze each stage of the malicious act using reverse engineering, allowing them to evaluate the harm level, spreading speed, removal process, and successful countermeasure to stop malicious codes from spreading [1]. It's worth noting that reverse engineering can be used by both security professionals and malicious code developers. These programmers employ reverse engineering to find the weak points in malicious code and improve it. They often use anti-debugging and rootkits to build codes that can evade security experts' study. As a result, security experts are looking at ways to deal with these ever-evolving malicious codes. However, malicious code generation tools are finding their way around hacking forums and closed news, making malicious codes more available, and strategies to evade analysis are improving. As a result, it is difficult for security experts to protect against malicious code in a timely and efficient manner. To make matters worse, there are insufficient human resources to analyze malicious code, and analysis technology is not developing at the same rate as malicious code. As a result, security experts must think about rootkits in addition to malicious code. This paper is motivated by this, and it investigates rootkit techniques used in malicious code, as well as their countermeasures. Furthermore, sample rootkits are introduced and performed in order to investigate rootkit techniques in a more realistic and efficient manner.

II. LITERATURE SURVEY

Alhassan, et al [2] describes how rootkit detection mechanisms work. It shows that rootkit detection is based on Host based, Virtualization Based or External Observer Based. The paper also explains how profiling helps manual analysis to understand the attacker strategy and would in turn help to contain the attack. The paper talks about PoKeR(Profiler for Kernel Rootkits) which is capable of producing rootkit profiles which include the revelation of rootkit hooking behavior, targeted kernel objects, user levelimpacts. It is also capable of extracting rootkit code.

Kruegel, et al [3] describes a static analysis-based technique for detecting instruction sequences that are indicative of rootkits. Informal behavioral requirements describe common instruction sequences, such as data transfer operations that write to unauthorized kernel memory areas. Symbolic execution is then used to simulate the kernel module's execution in order to find instructions that meet these requirements. This approach allows for the detection of malicious actions before a module is loaded into the kernel, as well as the operation of closed source components such as proprietary drivers.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 5, May 2021 ||

DOI:10.15680/IJIRSET.2021.1005014

Liu, et al [4] proposes an improved method (software) to detect the rootkit, which is called X-Anti. X-Anti is based on the overall rootkit detection program and provides the detection function of five rootkit hidden elements, such as files, registration table, process, driver modules, and ports, etc. X-Anti can be used by many operating systems such as Windows XP (normal, sp1, sp2 and sp3), windows 2003 (normal, sp1, sp2), Vista and Window 7 (under certain conditions).

Bowman, et al [5] describes that it is very easy to find the source code for rootkit on the internet and the latter was able to get complete, stealthy, access, and control of targeted Linux and Windows based systems. The latter were not even programming experts but still were successful in implementing the rootkit in the machine which shows rootkit is extremely dangerous and can easily be installed in victim machines through social engineering.

Joy, et al [6] implies that rootkit is the type of "Trojan Horse" that resides in OS. The paper describes that they are the most difficult attack to detect due to their self-concealment actions and administrative level rights. Researchers are searching for methods that require less prior knowledge, have a low overhead, and are accurate. The detection mechanism may be categorized as host-based, virtualization-based, or external observer-based depending on where the detection module is located. The virtualization-based approach is the most commonly used detection method, but it has some drawbacks in terms of implementation.

Nerenberg& Daniel [7] has shown that the rootkit techniques and techniques of detection of the rootkit are important. Many of the latest technologies were illustrated for every stealth technique and detection class. A graphical representation called an attack tree can be used to examine rootkit stealth techniques. In this, the attack tree and its counterpart, the defense tree, show which techniques touch the tree's foundation and that each rootkit category has to be addressed to defend itself successfully against all rootkits. It has been demonstrated experimentally that a rootkit can find other rootkits successfully. This study examined how rootkits are hidden and how to detect them. These hiding techniques were used to create an attack tree from which deficiencies in current detection techniques can be identified. Categorizations of detection have also been refined.

XiongweiXie&Weichao Wang [8] Proposes a new rootkit detection mechanism for virtual machines based on deep information extraction and reconstruction at the hypervisor level in this paper. First, the hypervisor will reconstruct the semantic view of the VM's memory. Cross-verification among the various components of the recon-structured view the hypervisor can detect hidden information and mismatch among different active modules in the VM. This experiment results show that the proposed approach is practical and effective in rootkit detection.

Bickford, et al [9] implies that rootkits avoid detection by infiltrating the operating system, allowing them to bypass user-space detection tools and operate invisibly for extended periods of time. This paper demonstrated how kernel-level rootkits can exploit smartphone operating systems, frequently with serious social consequences. Because of the popularity of the mobile platform, attackers have increasingly begun to develop and deploy viruses and worms.

Win, et al [10] present here a rootkit and malware detection system to protect the virtualization infrastructure in cloud environments against cyber-attacks. It uses system calling body hazing and the use of SVM in conjunction with VMI to overcome limitations associated with the current virtualization security approaches. By using a secure in-VM monitor, it is guaranteed that the internal VM guest status can be obtained accurately without compromising it while using an offline SVM classifier in the remote monitor enables a quick classification of attacks.

Wangen [11] presents a taxonomy of APT mechanisms and cyber-espionage consequences has been presented. It shows clearly that cyber spying knows no borders. The taxonomy highlights various aspects of a growing problem. Technically, we see APT behaviors, infection rates, destination platforms and patterns of general attacks. We see the probable targets and origins of attacks on a geographic level, for instance the historically unstable Near East is also an attractive cyberspace target. The point is that a lot of information can be gathered that compiles such data sets. We have compiled and classified with our taxonomy much of the information that exists about major cyber spy attacks, including malware. The knowledge derived from this taxonomy will help us better defend against cyber spying and help scholars to learn and teach about this topic. However, the data set compiled in this paper needs to be extended further and more research is therefore needed. To construct a complete open-scale dataset, research and the professional community will need collaboration that could further facilitate the geopolitical and/or technical analysis and synthesis of malware's role in cyber spying.

III. ROOTKIT DEFINITION

A rootkit is a piece of software designed to hide the existence of its related programs from inspection and detection while retaining privileged access to a target device. It's worth noting that modern malwares often use rootkit techniques to mount and unlock themselves without being detected, causing significant harm to victims' computers

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 5, May 2021 ||

DOI:10.15680/IJIRSET.2021.1005014

[12]. The rootkit may alter the operating system (OS) or existing applications, some of which may otherwise be used for malware detection and analysis, once it is installed. Since it can subvert security or antivirus programs whose purpose is to locate it, it is extremely difficult to detect the rootkit and its associated programs. As a result, detecting and dealing with the rootkit becomes a significant challenge. Normally, each OS has two modes: a user mode (ring 3) and a kernel mode (ring 0). Actual codes are executed in kernel mode to perform the OS's core tasks, while in user mode, those codes are executed indirectly through an Application Programming Interface (API). In other words, the API is used in user-mode to assist in accessing kernel-mode rather than explicitly processing kernel operations. A rootkit normally hides itself in the user-mode and kernel-mode access processes.

IV. USER-LEVEL ROOTKIT

Since user-level rootkits have no control over the kernel structure, user-level rootkits use API hooking to cover themselves. API Hooking, in more detail, aids these rootkits in their concealment by gaining access to the program's memory address space and manipulating its data. The key downside of user-level rootkits is that they are quickly identified by rootkit detection software. On Windows, there are three subsystems: Win32, POSIX, and OS/2. Each subsystem is made up of APIs, which enable a process to ask its operating system to perform the operations it needs. As a result, a user-level rootkit can exploit APIs (API hooking) to alter the OS's standard operations at will. Import Address Table (IAT) hooking and inline hooking are two types of API hooking. In IAT hooking, a hacker attempts to overwrite an original function address in IAT with the rootkit's address. The rootkit is then executed if the original function is called. Unlike IAT hooking, inline hooking aims to change the function code of a victim program at will, thereby missing the function address. The rootkit code should be inserted into another process' address space to hook the API. The Dynamic-Link Library (DLL) Injection method is an example of this.

4.1 DLL INJECTION

DLL injection is a technique for inserting rootkit or malicious code into the address space of another process. A hacker can change the execution flow of a program's API calls after injecting a malicious DLL into it. This approach can be divided into three categories: using the registry, using the hooking tool, and using the remote thread.

• DLL Injection using registry

A hacker can insert a malicious DLL into Windows NT/2000/XP/2003 by changing the registry value HKEY_LOCAL_MACHINE\ Software\ Microsoft\ WindowsNt\ CurrentVersion\ Windows\ AppInit_DLLs. Since the modified registry value must be added to the device, rebooting is needed to use this injection. The machine would no longer need to boot for the newly started processes until the registry update is reflected. This approach is obviously simple to implement, but it does necessitate at least one reboot. As a result, the security expert will simply check the registry any time the computer boots up to avoid DLL injection.

• DLL injection using window hooking function

Application programs based on events are running in a windows OS. The SetWindowsHookEx [14] feature allows an application to install a user-defined Hook procedure in the hook chain to handle an event in a special manner. This function will be called with an event and a hook procedure, as demonstrated in the function prototype. Then, every time the event is loaded into the function's virtual memory, the hook procedure is called. DLL injections need to be monitored with windows or remote thread in real time as they are operated without rebooting immediately. For this purpose, we can use the PDESKTOPINFO pDeskInfo structure in the structure THREADINFO which has desktop information in case the window hooking function is used. In the DESKTOPINFO structure, PHOOK [13] aphkstart is the array of the global hooking list. Because the HOOK structure has information about the type of hooking event, the simple search can tell if the global hooking is set.

V. KERNEL-LEVEL ROOTKIT

The kernel-level rootkit is difficult to identify because it performs hidden behaviors by hooking the native API and runs at the same level as the anti-rootkit program. In Kernel mode, the most frequently used hooking methods are the System Service Descriptor table (SSDT), the Interrupt Descriptor Table (IDT) and an important function of the device driver object. As the anti-rootkit software allows for modified execution codes, several attackers are using the DKOM technique to modify kernel level information [17].

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 5, May 2021 ||

DOI:10.15680/IJIRSET.2021.1005014

5.1 System Service Dispatch Table (SSDT) Hooking

Windows from Microsoft supports Win32, POSIX and OS/2 subsystems. Firstly, the System Service Dispatch Table is a specific kernel structure for these subsystems (SSDT). This table keeps the methodology memory addresses in the order of system calls. In addition, it is possible to manage parameter dimensions used in each system service factions in the Systems Parameter Table (SSPT) and, via the ZwQuerySystemsInformation feature, to obtain various data types in a Microsoft Windows. This function lists running processes, so a rootkit can hide one process by changing the NwQuerySystemInformation address. The rootkit hides to evade detection by implication of the SSDT hook. In this attack, we can keep a white list of forges/modulation detection tables, including all the SSDT tables.

5.2 Interrupt Descriptor Table (IDT) Hooking

A system table is used for the interrupt handle by the Interrupt Descriptor Table (IDT). The software or hardware interrupt, and one of the typical interrupts is a keyboard interrupt. An OS runs the keyboard interrupt when the user types in a keyboard by calling the interrupt handler. You can use the sidt directions for collecting information and IDT location. Therefore, an attacker can hook using this instruction by collecting IDT information. Please note that this technique can leak the actual keyboard data. By constructing a white list, including all IDT tables, we can detect forge/modulation [18].

VI. CONCLUSION

The rootkit techniques were briefly discussed in this article, as well as the countermeasures. The sample rootkits, which have been specially implemented to demonstrate how they work, are used to power such a survey. It will be important to examine the most recent rootkit technologies in newly emerging computing platforms such as smartphones, cloud computing, and so on in future research.

REFERENCES

[1] M.Kim, Sungkwan, et al. "A Brief Survey on Rootkit Techniques in Malicious Codes." *J. Internet Serv. Inf. Secur.* 2.3/4 (2012): 134-147.
[2] Alhassan, J. K., S. O. Subairu, and S. Misra. "Evaluating Capabilities Of Rootkits Tools."
[3] Kruegel, Christopher, William Robertson, and Giovanni Vigna. "Detecting kernel-level rootkits through binary analysis." *20th Annual Computer Security Applications Conference*. IEEE, 2004.
[4] Liu, Leian, et al. "Research and design of rootkit detection method." *Physics Procedia* 33 (2012): 852-857.
[5] Bowman, Michael, Heath D. Brown, and Paul Pitt. "An undergraduate rootkit research project: How available? How hard? How dangerous?." *Proceedings of the 4th annual conference on Information security curriculum development. 2007.*
[6] Joy, Jestin, Anita John, and James Joy. "Rootkit detection mechanism: a survey." *International Conference on Parallel Distributed Computing Technologies and Applications*. Springer, Berlin, Heidelberg, 2011.
[7] Nerenberg, Daniel D. *A study of rootkit stealth techniques and associated detection methods*. AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH GRADUATE SCHOOL OF ENGINEERING AND MANAGEMENT, 2007.
[8] Xie, Xiongwei, and Weichao Wang. "Rootkit detection on virtual machines through deep information extraction at hypervisor-level." *2013 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2013.
[9] Bickford, Jeffrey, et al. "Rootkits on smart phones: attacks, implications and opportunities." *Proceedings of the eleventh workshop on mobile computing systems & applications*. 2010.
[10] Win, Thu Yein, HuagloryTianfield, and Quentin Mair. "Detection of malware and kernel-level rootkits in cloud computing environments." *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*. IEEE, 2015.
[11] Wangen, Gaute. "The role of malware in reported cyber espionage: a review of the impact and mechanism." *Information* 6.2 (2015): 183-211.
[12] URL: Rootkit. Veracode. https://www.veracode.com/security/rootkit accessed on 26th April 2021
[13] URL: Hooks Overview. Microsoft Documentation https://docs.microsoft.com/en-us/windows/win32/winmsg/about-hooks accessed on 26th April 2021
[14] URL: SetWindowsHookEx function. Microsoft Documentation https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowshookexa accessed on 26th April 2021
[15] URL: Rootkit: Kernel Mode. Infosec Institute https://resources.infosecinstitute.com/topic/rootkits-user-mode-kernel-mode-part-2/ accessed on 26th April 2021
[16] URL: Rootkit. Heimdal Security https://heimdalsecurity.com/blog/rootkit/ accessed on 26th April 2021

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 5, May 2021 ||

DOI:10.15680/IJIRSET.2021.1005014

[17] URL: Rootkit: Kernel Mode. ScienceDirect https://www.sciencedirect.com/topics/computer-science/kernel-rootkits accessed on 26th April 2021

[18] URL: Hooking IDT. Infosec Institute. https://resources.infosecinstitute.com/topic/hooking-idt/ accessed on 26th April 2021

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.512

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर
NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details