MALWARE ANALYSIS

LITERATURE REVIEW

AIR UNIVERSITY

HIDAYAT UR REHMAN 211833

ENVIRONMENT-AWARE MALWARE (WHICH DETECTS BEING EXECUTED IN A SANDBOX)

LITERATURE REVIEW

Dynamic analysis on a virtual computer is a frequent technique for analyzing possibly dangerous software. The behavior of malware when it is executed in a VM is therefore modified by the virus's designers via various approaches. But how exactly do they accomplish it?

Dynamic analysis is a popular method for analyzing software that may be dangerous. The binary is run in an analysis environment—typically a Virtual Machine (VM)—and the behavior it exhibits while running in the system is examined. Knowing this, malware authors work to prevent it by concealing their dangerous purpose during analysis. The virus must determine that it is operating within a VM in order to be able to achieve this. For this, there are many different detection techniques, which we will go over in more detail. In light of this, we also examined the use of VM detection techniques on 50,000 [1] malware samples. Low-level methods such as examining hardware and network information, malware can also use high-level methods that utilize the Windows API to detect whether it is running in a virtual machine. For example, malware can use the RegOpenKey function to search for specific registry keys that may indicate the presence of a virtual machine. This is just one example of how malware can use the Windows API to detect its environment. Malware authors are always coming up with new ways to detect their environment, and as a result, it can be challenging for antivirus software and other security tools to keep update. For this reason, it's important to use a combination of different techniques and approaches to detect and protect against malware. Another trend that malware authors have been using is to utilize Windows Management Instrumentation (WMI) classes to retrieve information about the system. The MSAcpi_Thermal Zone Temperature class can be used to check for the presence of a virtual machine, by examining the Current Temperature property. If this property is not supported, it is likely that the malware is running in a virtual machine, as virtual machines do not have hardware that can be used to measure temperature. This is just another example of how malware can use WMI classes to detect its environment. Some other methods that can be used to check for system artifacts indicating the presence of a hypervisor or other virtualization software[2]. For example, malware can use WMI classes to check for the presence of virtualization tools or specific registry keys that may indicate the presence of a virtual machine. Some of the technique discussed in [3] and [4] are listed following.

Hardware Information: Malware can collect information about the hardware it is running on, such as the processor type, the available memory, and the attached peripherals. This information can be used to determine whether the malware is running on a physical machine or a virtual machine. Network Information: Malware can collect information about the network it is connected to, such as the IP address, the MAC address, and the DNS server. This information can be used to determine whether the malware is running on a physical machine or a virtual machine. Process Information: Malware can enumerate the running processes on the system and examine their properties, such as the process name, the command line arguments, and the loaded DLLs. This information can be used to determine whether certain security tools or sandbox environments are running on the system, which may indicate that the malware is running in a virtual machine. File System Information: Malware can examine the file system to look for specific files or directories that may indicate a particular environment. For example, malware may look for the presence of virtualization tools, which may indicate that it is running in a virtual machine. Registry Information: Malware can access the registry to look for specific keys or values that may indicate a particular environment. For example, malware may look for the presence of debugging tools or sandbox environments, which may indicate that it is running in a virtual machine. A mechanism of communication between the guest OS and the Virtual PC VMM is offered by the invalid instruction 0x0f,0x3f. While Bytes 3 and 4 can include a variety of various values, each of which represents a call to a separate VMM service, the values used are most prevalent ones (0x07 and 0x0b) seen in malware that is aware of Virtual PCs (VPCs). It is challenging to determine the fraction of malware that is VM-aware. Static analysis techniques cannot be relied upon to accurately measure this fraction since they are readily thwarted by code that has been obfuscated or encrypted. It would take too long to measure across a statistically relevant set of samples when utilizing DSD-Tracer for dynamic analysis (e.g. to achieve low margin of error and high level of certainty)[5]. A decent approximation is provided by the mix of static and dynamic methods, enabling the reader to make choices based on the information in the article. They created DSD-Tracer, a system that can accurately analyses various virtual machine identification techniques in a software while under time constraints. In [5] they calculated that 2.13% of the samples overall are aware of virtual machines. Although this figure is not as big as some have suggested, it is still a sizeable figure that must be considered when performing analysis on virtual computers. Additionally, it demonstrates the need for precautions to be taken while developing VM-based automated analysis systems. In order to find out whether VMware is present, this method employs "backdoor" communication over port 0x5658 (VX) from VMware [6] . Old harmful software is constrained since it doesn't clearly outperform intrusion detection technologies that are integrated into a target machine's operating system. They showed how attackers might have a definite advantage over intrusion detection systems operating on a target OS in our study. They looked at how VMBRs, which employ VMMs to provide attackers qualitatively greater control over compromised systems, are designed and put into practice. They demonstrated how attackers may take advantage of this advantage to easily create malicious services that serve a variety of purposes while remaining entirely hidden from the victim system. By putting two proof-of-concept VMBRs into use, they assessed this new malware threat. They developed four sample malicious services on target Windows XP and Linux computers using our proof-of-concept VMBRs [7]. Last but not least, VMBRs [7] implement the overall concept of adding a new layer to an already-existing system. Other uses of this concept include stacking file systems [8], virtual machines [9], and maintaining system compatibility by leaving network firewalls alone. Sandboxes are isolated environments that allow malware to be safely analyzed without damaging the host system. By observing the events that occur during the execution of the malware, analysts can gain detailed information about the malware's capabilities and behavior, including its interactions with the network, file system, registry, and other system resources. Dynamic analysis can be a useful tool for understanding how malware operates and for identifying its features and capabilities. It can also help analysts to identify the actions that the malware takes when it is executed, such as the files it creates or modifies, the network connections it establishes, or the registry keys it accesses. This information can be used to develop detection and defense strategies against the malware [10]. In contrast to conventional methods, iterative pattern mining was employed by the authors to identify malware. This method was based on the supposition that malware performs repetitive actions on data sequences, such as running infestations or running loops that carry out a decryption/encryption process. The total procedure is divided into five phases by the study's authors. By executing these samples in a controlled virtual environment, such as a lab, they were able to record PE interaction with operating system APIs after collecting malware samples in the first stage. They employ VMWARE and Qemu [11]. The study in [12] was performed using dynamic analysis tools like Capture-Bat, Regshot, APATE DNS, PEID, PE Explorer, or Sysinternal, and sophisticated dynamic analysis tools like Virmon, Cuckoo, and WINAPIOverride32 were utilized to mimic the behavior of malware. Agent-based and Agentless Simulator Systems and discover that Agentless Sandbox is more effective at recognizing complex malware, which bypasses or crashes itself on finding itself being identified by Sandbox Agent [12]. A thorough examination of dynamic analysis evasion on various platforms has been done by the authors in [13]. The issue of manual dynamic analysis evasion is just briefly touched upon because their emphasis is on automated dynamic analysis. Other studies, such as [14, 15], only lightly examined analysis evasion and gave no comprehensive summary of malware dynamic analysis evasion.

REFERENCES

[1] Kemkes, P. (2020, May 8). VM detection methods in malware. VM Detection Methods in Malware. Retrieved December 25, 2022, from https://www.gdatasoftware.com/blog/2020/05/36068-current-use-of-virtual-machine-detection-methods

[2] Andrea Fortuna. "Malware VM detection techniques evolving: an analysis of GravityRAT". 2018. https://www.andreafortuna.org/2018/05/21/malware-vm-detection-techniques-evolving-an-analysis-of-gravityrat/

[3] Bursztein, E., Seifert, J.-P., Jahanian, F., & Savola, R. (2010). Understanding botnet marketing. In Proceedings of the 20th ACM Conference on Computer and Communications Security (pp. 301-312). ACM.

[4] Chen, Y., & Cui, Y. (2011). A systematic study of botnet marketing. In Proceedings of the 20th ACM SIGSAC Conference on Computer and Communications Security (pp. 791-802). ACM

[5] Lau, B., Svajcer, V. Measuring virtual machine detection in malware using DSD tracer. J Comput Virol 6, 181–195 (2010). https://doi.org/10.1007/s11416-008-0096-y

[6] Kato, K.: VMWare Back. http://chitchat.at.infoseek.co.jp/vmware/backdoor.html (2003)

[7] S. T. King and P. M. Chen, "SubVirt: implementing malware with virtual machines," 2006 IEEE Symposium on Security and Privacy (S&P'06), 2006, pp. 14 pp.-327, doi: 10.1109/SP.2006.38.

[8] J. S. Heidemann and G. J. Popek. File-system development with stackable layers. ACM Transactions on Computer Systems, 12(1):58–89, February 1994.

[9] B. Ford, M. Hibler, J. Lepreau, P. Tullmann, G. Back, and S. Clawson. Microkernels Meet Recursive Virtual Machines. In Proceedings of the 1996 Symposium on Operating Systems

[10] Sujyothi, A., & Acharya, S. (2017). Dynamic Malware Analysis and Detection in Virtual Environment. International Journal of Modern Education & Computer Science, 9(3).

[11] M. Ahmadi, "Malware detection by behavioural sequential patterns."

[12] M. Ali, S. Shiaeles, M. Papadaki and B. V. Ghita, "Agent-based Vs Agent-less Sandbox for Dynamic Behavioral Analysis," 2018 Global Information Infrastructure and Networking Symposium (GIIS), 2018, pp. 1-5, doi: 10.1109/GIIS.2018.8635598.

[13] Alexei Bulazel and Bülent Yener. 2017. A survey on automated dynamic malware analysis evasion and counterevasion: PC, mobile, and web. In Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium. ACM,2

[14] Yuxin Gao, Zexin Lu, and Yuqing Luo. 2014. Survey on malware anti-analysis. In 5th International Conference on Intelligent Control and Information Processing (ICICIP '14). IEEE, 270–275.

[15] Jonathan A. P. Marpaung, Mangal Sain, and Hoon-Jae Lee. 2012. Survey on malware evasion techniques: State of the art and challenges. In 2012 14th International Conference on Advanced Communication Technology (ICACT'12). IEEE, 744–749.