International Journal for Electronic Crime Investigation

ISSN: 2522-3429 (Print)
ISSN: 2616-6003 (Online)

DOI: https://doi.org/10.54692/ijeci.2024.0802196

Vol. 8 issue 2 Apr-Jun 2024

Modern Malware Evasion and Bypass Strategies Against Contemporary Antivirus Software

Rabia Mehmood, Waleed Ahmed, Rehan Riaz, Waleed Hassan Awan

Department of Computer Sciences, COMSATS University, Lahore

Corresponding author: rabiamehmoodciit@gmail.com

Received: March 30, 2024; **Accepted:** May 09, 2024; **Published:** June 14, 2024

ABSTRACT

Antivirus software is the first line of defense against increasingly sophisticated cyberthreats, which emphasizes the need to understand and address its limitations. This study meticulously examines the complicated terrain of malware evasion and bypass strategies to shed light on the accessibility and ease of use of tools employed by hackers. A major cybersecurity concern is the ongoing growth of malware evasion and bypass techniques. The primary goal is to enlighten users about the constantly evolving threats and arm them with the information necessary to appropriately fortify their digital defenses. Antivirus software is the first line of defense against increasingly sophisticated cyberthreats, which emphasizes the need to understand and address its limitations. This study meticulously examines the complicated terrain of malware evasion and bypass strategies to shed light on the accessibility and ease of use of tools employed by hackers. A major cybersecurity concern is the ongoing growth of malware evasion and bypass techniques. The primary goal is to enlighten users about the constantly evolving threats and arm them with the information necessary to appropriately fortify their digital defenses. When updated regularly, contemporary antiviral software demonstrates strong resistance against a variety of tried-and-true approaches. This document provides a detailed discussion of packing, obfuscators, protectors, reflective DLL injection, remote process memory injection, process hollowing, and inline hooking. The research then goes into greater detail on various evasion techniques, including defensive evasion by making direct system calls and advanced evasion strategies, demonstrating the adaptability of malware creators in eluding antivirus and endpoint detection and response systems.

Keywords: Malware evasion, bypass, Cybersecurity, inline hooking, cyberthreats, digital defenses.

1. INTRODUCTION

The internet is now freely accessible to people all over the world, who utilize it for information collecting, communication, and enjoyment. Nevertheless, by utilizing harmful software and having internet access, hackers can jeopardize user privacy and critical data. As a protection against such attacks, users routinely install antivirus software, which aims to reduce the chance of device infection. Even well-known tools like web browsers can get infected when data are downloaded from unknown sources. For this reason, downloading content from reputable and verified sources is essential. Antivirus programs often alert the user when a suspicious item is downloaded and quarantine the infected file so that it can be cleaned up or removed. Users need to trust antivirus software, as the instrument and the internet is now freely accessible to people from all over the world, who utilize it for information collecting, communication, and enjoyment [1]. To counter new threats, antivirus software manufacturers constantly create new security techniques and update signature databases. The growing number of new infections suggests, therefore, that the security measures in place today might not always be adequate. To access devices, hackers employ a variety of evasion and bypassing strategies. To counter new threats, antivirus software manufacturers constantly create new security techniques and update signature databases. Nonetheless, the growing quantity of novel viruses suggests that the defense mechanisms in place today might not always be adequate [2]. This paper explains typical bypassing methods and tools that most antivirus software should be able to identify, but it also demonstrates how sophisticated attack chains that combine evasion strategies can get past contemporary, widely used antivirus software.

2. LITERATURE REVIEW

Installing antivirus software is seen as a crucial first step in safeguarding one's privacy on the internet. This literature study, however, explores the shortcomings of these products, emphasizing the ease of use and accessibility of techniques used by hackers to get around antivirus software. There is a significant association between the popularity and antiquity of evasion tools and their effectiveness, even though modern antivirus software that receives frequent updates works well against them. Interestingly, the study highlights default configuration weaknesses, showing that even the most recent antivirus software can be tricked by small changes to well-established evasion strategies. The study emphasizes the need for ongoing watchfulness since hackers use easily available resources to take advantage of potential vulnerabilities. The literature's ultimate goal is to increase user awareness of the hazards related to cybersecurity by advising them to stay vigilant and knowledgeable about the latest developments in digital threats.

Extending the research, the authors contrasted in a later paper the efficacy of antivirus software bypassing techniques on the Windows operating system with Kalogranis' work. In order to expand on their research, the authors included a new antivirus bypass tool dubbed The Fat Rat [3] , replicated the tests using the tools used by Kalogranis, and utilized a payload created with Metasploit. Shellter and Veil-Evasion were unable to get past security. Of the six antivirus applications that were employed, The Fat Rat was able to bypass one (PeCloak.py 4) [4], whereas Avet was able to bypass five.

The developers chose to limit their testing to Bitdefender after reading an analysis of the antivirus software in another study [5], which ranked Bitdefender as one of the top options. The target PC was able to access the Remote Access Trojan (RAT) malware through the use of the Apache server. The authors examined nine different antivirus bypass methods, taking into account whether the antivirus program would be able to detect RAT as well as whether it would be able to prevent the triggered Meterpreter session that RAT activated. As a fraction of the total number of ways for each tool, the effectiveness of these tools was displayed [6].

This paper presents a new approach to return-oriented programming (ROP) based code obfuscation. The two main aspects of ROP automated analysis and creation of ROP chains for a given code and the repurposing of valid code as ROP gadgets pose problems to standard malware research. The developed program, ROP Injector, uses executable code to patch the ROP chain and convert shellcode to its ROP equivalent. Experimental results on VirusTotals show that ROP Injector can bypass nearly every antivirus program, demonstrating the efficacy of ROP in obfuscating code. This study highlights the need for improved cybersecurity measures by highlighting the possible threat posed by ROP in cyberattack campaigns. The writers concentrated on malware that can change its code on the fly to avoid detection, known as polymorphic malware. This method entails developing several malware variations, each with a unique code signature. Upon execution, the malware randomly chooses and runs one of the variations. Because each form of the malware has a different code signature, this makes it harder for antivirus software to detect the malware [7].

3. METHODOLOGY

Malware evasion refers to strategies used to evade security system detection. This can involve using encryption to conceal dangerous payloads, polymorphic code that alters its appearance, and taking advantage of security software flaws. Avoiding detection frequently necessitates constant adjustment to security solutions' counter measures. Malware evasion can be on desk or in memory.

3.1. Malware Evasion On-Desk

3.1.1. Packing

Malware is packed similarly to a compressed file, including new instructions and a larger file size to evade "signature-based detection" [8].

3.1.2. Obfuscators

It obfuscates the blacklisted functions, such as Virtual Alloc, Virtual Protect, and more, so that they can avoid "Heuristic Detection," which makes it difficult for the program to understand the instructions from antivirus software.

3.1.3. Protectors

Although it complicates the malware's reverse engineering process, the Protectors app[4] is a regular one that wasn't intended for use in evasion, but it still has its uses.

3.2. Process hollowing

We create a fictitious process that consumes space, pause it, modify its content to match our payload, and then restart the process with his updated instructions and content. A technique called "inline hooking" allows you to change a process's code while it's still executing in memory. Redirecting function calls from the original code to a new location in memory accomplishes this. Although there are other approaches, these are the well-known ones that are employed. In the current digital era, cyber-attacks are a constantly changing concern. It's critical to stay one step ahead of attackers who are always coming up with new ways to get around established defenses. This research outlines the strategies employed by these attackers, with a particular emphasis on how they evade Endpoint Detection and Response systems. Certain malware can successfully evade detection by employing strategies like the usage of syscalls. These strategies go beyond the first infiltration phase. Attackers use sophisticated tactics like process injection and DLL hijacking to keep control of the system after they have gained access. Regarding analysis, 'Dark Crystel RAT (DCrat)' is highlighted as a leading illustration of contemporary cyber risks. Examining this danger in depth gives readers a thorough grasp of the difficulties this type of malware poses by illuminating how it operates. This information serves as a tool and is not merely academic. Individuals, companies, and organizations can better prepare and safeguard their digital assets in an increasingly hostile cyber environment by being aware of these hazards [9]. Following are the techniques used:

3.2.1. Technique # 1: Defense Evasion Technique Using Direct Syscalls and Advanced Evasion Methods

In order to escape AV/EDR detection, this strategy entails creating a suite of tools that utilize direct syscalls, evade sandboxes, employ strong encryption, and change procedure names. It also describes how to circumvent security protections and generate memory snapshots using the well-known utility Dumpert, which makes use of direct syscalls [10]. Notably, Microsoft Defender identified Dumpert after it was created and utilized on the disk. This discovery prompted research into avoidance strategies for both static and dynamic scenarios. You need to have the basic knowledge of Native APIs and Windows API. On Windows, applications execute in user mode. Aggressors: conduct operations with Windows APIs. AV/EDR security solutions have no visibility beyond the native APIs that are present in ntdll. dll. Ever heard of malware that executes via

Windows API functions writeprocessmemory,createremotethread, and virtualallocex? These APIs connect to more ntdll dll API activities. Most of the operations in ntdll. dll contains instruction steps to start kernel system-level tasks. Because whenever the app is carrying out these actions, the AV/EDR tools hook into Native APIs and change the direction of the application which allows them to listen for malicious behavior on that one app. EDRs load their DLLs into the process memory at startup to monitor. Outflank created an amazing program that creates memory dumps by using straight syscalls. But because it's open-source, the majority of AV/EDRs have updated their signatures to support Dumpert. Instead of changing the signature, a different and more effective bypass technique was selected, with remarkable results. In order to avoid Dumpert's static analysis, in-memory execution is used. Although Dumpert's default method for creating memory dumps is through direct syscalls, an injector was also created to load Dumpert shellcode into a remote process. The same approaches that were previously mentioned are incorporated into this loader [11].

3.2.2. Technique # 2: Achieving Elevated Reverse Shells via DLL Hijacking and Mock Directories

The goal of this approach is to obtain a high-level privileged reverse shell by circumventing Windows UAC security features through the use of DLL Hijacking and Mock directories. The method, which security experts have identified, uses dummy files in conjunction with a simplified DLL hijacking procedure to get around UAC protections. Tests on Windows 10 were able to successfully disable the UAC security mechanism, raising concerns about how resistant Windows 11 is to similar tactics. Escalating privileges is usually the next step after gaining initial access, with objectives such as hash dumping or performing[18] privileged actions that enable lateral movement inside a network. Think about a domain user who uses a PC and is also the local administrator. In the event that this user is compromised by an attacker, there is an instantaneous push to elevate privileges in order to dump hashes and utilize that user's NTLM hashes for network authentication. But since there is already an elevated reverse shell in place and a privileged connection to the C2 server established, there is no need for this kind of escalation. This method is called Mock Directories [12].

In essence, a fake directory is a mimicked directory that can be identified by its trailing space. Consider the Windows trustworthy directory "C:\Windows\System32." The dummy equivalent would be "C:\Windows\System32," with the trailing space being the main distinction. Here, it's crucial to emphasize that Windows Explorer cannot be used to create mimic directories. PowerShell or the command prompt (cmd) must be used for creation. It is not possible to create "C:\Windows," however it is possible to set up "C:\Windows \System32."

Taskmgr. exe's integrity level was checked throughout the study. Taskmgr.exe is located in "C:\Windows\System32" and loads many DLL files when it runs. Attackers have the chance to use the DLL hijacking technique with this program [13]. This procedure "autoelevates" each DLL it introduces because of its high integrity level by design. It is This section explores the attack's mechanism, showing how an attacker may bypass Windows 11's UAC protections and acquire an administrator shell by using DLL hijacking and fake folders. This method's effectiveness was verified on Windows 11, even while Windows Defender was turned on.

Following the creation of the shellcode, a straightforward C++ program was developed to produce a DLL file. This program incorporated the previously generated shellcode.

The next step is creating a batch program that creates fictitious folders, copies a file to one of these fictitious directories, and tries to load the malicious DLL. There are a number of ways to use Mimikatz and avoid Windows Defender detection. On the C2 server [11] , user hashes were collected when Mimikatz was successfully launched. Numerous network-wide attacks may be carried out to authenticate users using these NTLM hashes [14].

3.2.3. Technique # 3: Direct System Calls for AV/EDR Evasion, User-Mode vs Kernel-Mode

A variety of techniques are employed by contemporary AVs and EDRs to do both static and dynamic analysis. They possible to use many executables in a DLL hijacking attack. "computerdefaults.exe" is the attack executable selected in this method. Attackers use these binaries to increase level of power in Windows, enabling them to perform DLL hijacking and change registry settings, among other things.

may look at a variety of signatures, including keys, hashes, and recognized strings, to find out if a file on disk is dangerous. Nevertheless, attackers have created a wide range of obfuscation techniques, rendering static analysis all but useless [15].

Dynamic/heuristic analysis is the primary emphasis of modern EDRs, which allows them to keep an eye on how each process behaves on the system and search for unusual activity. As a result, if malicious files have been disguised, this approach can download them and perhaps leave the EDR unnoticed[9]. However, as soon as the virus is activated, the EDR will recognize it and stop it. User-land hooks are used by the majority of AVs, EDRs, and sandboxes to monitor and intercept each user-land API call. They are unable to trace a technique that enters kernel mode and conducts a system call [16].

The fact that system call numbers differ between OS versions and occasionally even between service build numbers presents a problem. Nonetheless, the inmemory NTDLL module may be scanned to retrieve the syscall numbers using a library called inline syscall. The tricky part of this is that this module uses Windows API calls to retrieve the syscall number. These routines will not obtain the right number if an AV/EDR hooks them. Using Syswhispers is one alternate method that this blog discusses. By creating header/ASM files that implants can utilize to start direct system calls, SysWhispers helps in evasion [17].

CONCLUSION

Proactive defense and awareness are crucial in the face of constantly changing cyberthreats. This study highlights the need for a comprehensive and constantly evolving strategy towards cybersecurity through its discussion of inventive methods and procedures. Conventional defenses still have their place, but ongoing learning and adaptation are also necessary. This research seeks to provide people and organizations with the knowledge necessary to strengthen their digital defenses through its thorough examination. Let this research serve as a light for improved cyber resilience as we traverse this digital age.

REFERENCES

[1] U. Ahmed, J. C.-W. Lin, and G. Srivastava, "Mitigating Adversarial Evasion Attacks of Ransomware Using Ensemble Learning," *Computers and Electrical Engineering*, vol. 100, pp. 107-129, 2022.

[2] J. Cabrera-Arteaga, M. Monperrus, T. Toady, and B. Baudry, "WebAssembly Diversification for Malware Evasion," *Computers and Security*, vol. 131, pp. 103-119, 2023.

[3] J. Chen, C. Yuan, J. Li, D. Tian, R. Ma, and X. Jia, "ELAMD: An Ensemble Learning Framework for Adversarial Malware Defense," *Journal of Information Security and Applications*, vol. 75, pp. 105-134, 2023.

[4] S. Choi, T. Chang, S.-w. Yoon, and Y. Park, "Hybrid Emulation for Bypassing Anti-Reversing Techniques and Analyzing Malware," *The Journal of Supercomputing*, vol. 77, no. 1, pp. 471-497, 2021.

[5] S. Gold, "Advanced Evasion Techniques," *Network Security*, vol. 20, no. 1, pp. 16-19, 2011.

[6] H. Anand, A. Handa, N. Kumar, and S. K. Shukla, "Adversaries Strike Hard: Adversarial Attacks Against Malware Classifiers Using Dynamic API Calls as Features," *Cyber Security Cryptography and Machine Learning*, pp. 20-37, 2021.

[7] D. Li, S. Cui, Y. Li, J. Xu, F. Xiao, and S. Xu, "PAD: Towards Principled Adversarial Malware Detection Against Evasion Attacks," *IEEE Transactions on Dependable and Secure Computing*, pp. 1-16, 2023.

[8] X. Ling, L. Wu, J. Zhang, Z. Qu, W. Deng, X. Chen and Y. Qian, "Adversarial Attacks against Windows PE Malware Detection: A Survey of the State-of-the-Art," Computers and Security, vol. 128, p. 103134, 2023.

[9] H. Liu, W. Sun, N. Niu, and B. Wang, "MultiEvasion: Evasion Attacks Against Multiple Malware Detectors,"

IEEE Conference on Communications and Network Security (CNS), pp. 10-18, 2022.

[10] A. Monika and R. Eswari, “Prevention of Hidden Information Security Attacks by Neutralizing Stego-Malware,” *Computers and Electrical Engineering*, vol. 101, pp. 107-124, 2022.

[11] M. Noor, H. Abbas and W. Bin Shahid, “Countering Cyber Threats for Industrial Applications: An Automated Approach for Malware Evasion Detection and Analysis,” Journal of Network and Computer Applications, vol. 103, pp. 249-261, 2018a.

[12] M. Rhode, P. Burnap and A. Wedgbury, “Real-Time Malware Process Detection and Automated Process Killing,” *Security and Communication Networks*, vol. 2021, pp. 1-23, 2021.

[13] K. A. Roundy and B. P. Miller, “Binary-Code Obfuscations in Prevalent Packer Tools,” *ACM Computing Surveys*, vol. 46, no. 1, pp. 1-32, 2013.

[14] A. Sharma, B. B. Gupta, A. K. Singh, and V. K. Saraswat, “Orchestration of APT Malware Evasive Manoeuvers Employed for Eluding Anti-Virus and Sandbox Defense,” *Computers and Security*, vol. 115, pp. 102-132, 2022.

[15] R. S. Kunwar, “Malware Analysis of Backdoor Creator: Fatrat,” *International Journal of Cyber Security and Digital Forensics*, vol. 7, no. 1, pp. 72-79, 2018.

[16] T. Tsafrir, A. Cohen, E. Nir, and N. Nissim, “Efficient Feature Extraction Methodologies for Unknown MP4-Malware Detection Using Machine Learning Algorithms,” *Expert Systems with Applications*, vol. 219, p. 119-132, 2023.

[17] D. Waterson, “Managing Endpoints, the Weakest Link in the Security Chain,” *Network Security*, vol. 20, no. 8, pp. 9-13, 2020.