World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(Review Article)

Antivirus evasion techniques in modern malware

Bogdan Barchuk [1, *] and Kyrylo Volkov [2]

[1] *Chief Technology Officer at CQR Cybersecurity.*
[2] *Senior Penetration Tester.*

World Journal of Advanced Research and Reviews, 2025, 26(02), 3154-3166

Publication history: Received on 12 April 2025; revised on 19 May 2025; accepted on 21 May 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.26.2.1966

Abstract

Cybercriminals use new antivirus evasion techniques in their malware to continue operating in a system despite security programs. It explores how the newest malware uses obfuscation, packing, anti-debugging, and tampering with system security to bypass modern protection solutions at endpoints. The research points out that, attackers find ways to take advantage of weaknesses in antivirus heuristics, signature databases, and models that analyze behaviors. Hackers give additional focus to modifying programs, editing data in the hex format, dodging debuggers, and using file-loading tools. The research group also studied how measures like Windows' SmartScreen and SafeSEH prevent bad software from being started. Detection by today's antivirus solutions has improved, but it turns out that most evasion techniques can still work because they are flexible and mutable. As a result, organizations must always work on new methods of preventing attacks and stay informed about any threats that may arise.

Keywords: Malware Evasion; Code Obfuscation; Anti-Debugging; File Binding; Antivirus Detection; Runtime Packers

1. Introduction

Malware, short for malicious software, has undergone significant evolution since its inception, adapting continuously to counter emerging cybersecurity defenses. In their early forms, viruses only used straightforward tricks to spread and infect newly infected computers. When antivirus detection relied on signatures, malware began using harder-to-detect techniques. Companies have started using heuristic and behavioral analysis in their modern antivirus programs to deal with new threats. Even so, malware is evolving, using clever tricks that make it tough to detect for most security systems.

Making malware invisible is extremely important to those who develop it. By evading detection, malware can remain for a longer period in computers and execute its tasks without being detected. Usually, cyber criminals apply obfuscation, polymorphism, packing, anti-debugging, and anti-virtualization tactics. Consequently, it is now important for cybersecurity experts to come up with more flexible defense methods.

Being aware of the battle between malware makers and developers of antivirus software is necessary to improve cybersecurity measures. Because malware master's new ways to avoid being detected, studying trending threat methods and evasion techniques helps shape the future of security solutions (Sahay et al., 2019). Malware detection technologies must improve as cyber threats are becoming more advanced (Alenezi et al., 2020).

1.1. Overview

Current antivirus products rely on various mechanisms to spot and prevent malware viruses. This technique continues to depend on a database of signature patterns to detect and flag malware-infected data. Still, this way of handling threats is not enough to deal with modern and updated threats. For this reason, modern antiviruses can spot suspicious routines

* Corresponding author: Bogdan Barchuk

in code, even if they have not identified the particular malware before. Also, real-time logs are studied to identify malware through the way it is active in the system. Thanks to machine learning and artificial intelligence, it is easier for systems to spot details that could reveal the presence of malware.

Still, in spite of these inventions, criminals find ways to get their malware past antivirus software. Some hackers disguise their malicious code using techniques called code obfuscation and packing to avoid being discovered. Malware checks for special environments that are used for analysis, and as a result, it stops or changes what it does when such environments are found. Further, malware can evade detection by ignoring rules given to them in test environments and acting normally.

Learning about each method and its evasion is necessary to improve cyber protection. Because attackers keep improving their tactics and malware, antimalware specialists are always coming up with better ways to stop them (Samociuk, 2023). To face new viruses and protect against them effectively, we must keep updating antivirus software (Rohith & Kaur, 2021).

1.2. Problem Statement

Antivirus products have a hard time coping with the rapid development of malware. Most old antivirus software has difficulty finding malware that is not yet known or has been changed extensively. When malware authors use techniques to make their code harder to detect, like obfuscation, packing, anti-debugging, and sandbox evasion, ordinary antivirus can no longer stop them. With there diverse evasion methods, malware becomes harder to identify, keeps working longer, and causes even greater harm. Moreover, the emergence of malware that can change its code structure makes it even harder to find. The complexity of these dangers pushes for the use of more innovative and flexible security systems. The evolving nature of cyber-attacks means that current endpoint protection must also improve to stop them from working. Should improvements in antivirus technology stop, the chance of successful cyber-attacks and security breaches dealing severe damage to individuals, organizations, and important systems would grow.

1.3. Objectives

I am researching the innovative ways that malware can sneak by advanced virus protection. The main goal is to realize how malware evades usual means of detection using various sophisticated approaches like making itself hard to understand or understand and compressing itself. Researchers examine these strategies to find out if they affect the effectiveness of today's antivirus and endpoint protection tools. It is also necessary to check whether security software can recognize and block simple forms of evasive malware. The research will also look into new ways malware hides from antivirus software and how these solutions are amended. To sum up, the goal is to equip experts with recommendations for making cybersecurity measures more effective. It is helpful for better detecting malware and minimizing the risk of stealthy attacks today.

1.4. Scope and Significance

This study examines the use of software techniques by malware to avoid being caught by antivirus software. Only software-related threats are included, since hardware-based attacks are not evaluated. The paper studies both basic and advanced evasion techniques, among them are code obfuscation, packing, anti-debugging, anti-virtualization, and sandbox evasion. The reason this study is significant is that it can aid cybersecurity experts as they work on improving malware detection tools. Realizing the different evasion strategies allows developers to improve their protection solutions for endpoints. Besides, the study helps cybersecurity experts understand current risks and suggests where improvements can be made. Today's heavy use of digital systems means bolstering antivirus against evasive malware is key to keeping sensitive data secure and the system stable.

2. Literature Review

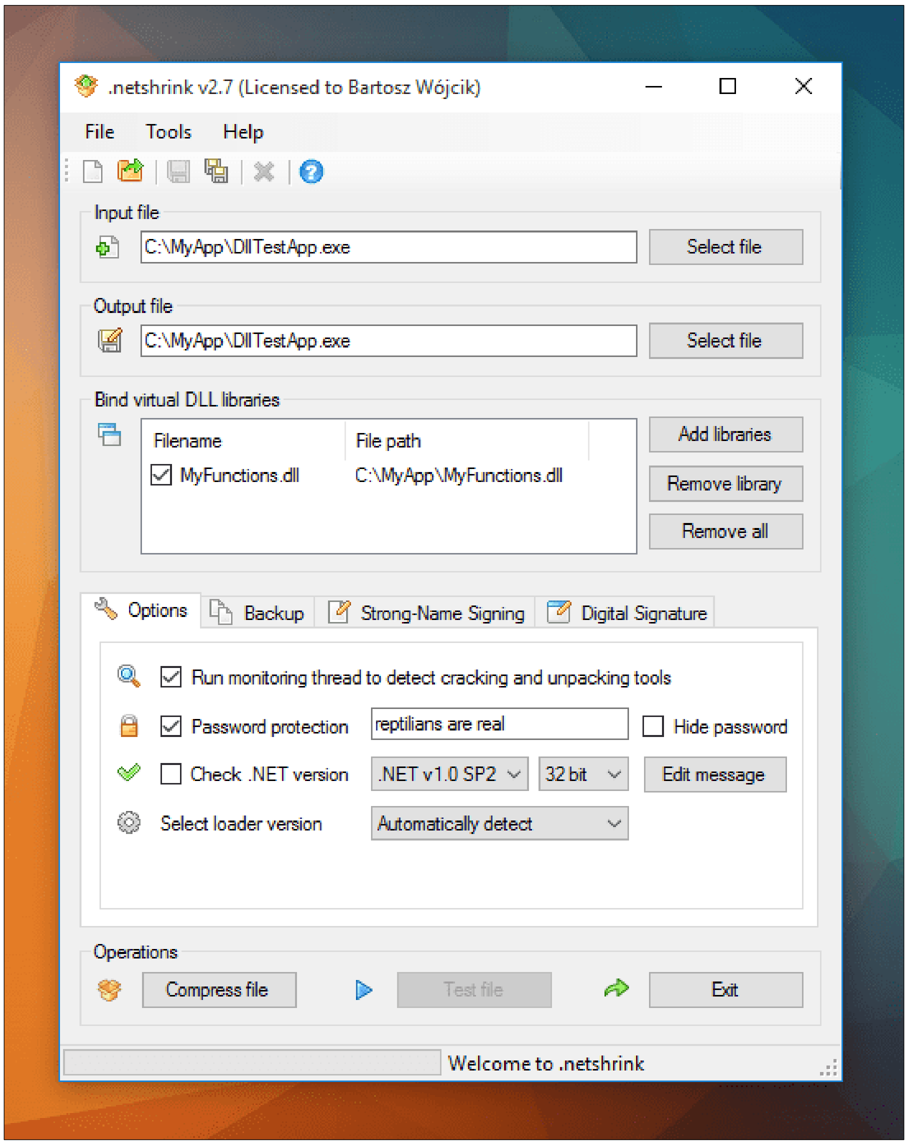
2.1. Basic Information on Evasion and Obfuscation

Developers of malware implement different ways to make it difficult for antivirus programs to spot. Fully UnDetectable (FUD) malware means malware created to go undetected by common antivirus programs. Many organizations use crypting and obfuscation to gain FUD status. The use of crypting covers up the malware by making it unclear, and obfuscation is applied to make its code harder to understand. Because of these tactics, antivirus software has difficulty identifying malicious programs using recognized or suspicious patterns.

If used properly, anti-debugging and anti-VM strategies can make it difficult for both researchers and automated systems to analyze a threat. To avoid being studied, anti-debugging methods find signs of debugging tools and sometimes block the analyst or abort the malware's execution. They also work by finding out if a virtual machine is being used to run malware, as this is a frequent method in dynamic analysis. The fewer places malware visits, the less likely it is to be spotted by a security system.

Anti-virtualization and anti-debugging methods have been reexamined and made stronger, such as by modifying system hooks that analysts rely on to spy on what malware does. This kind of malware is able to avoid even rigorous monitoring methods, creating additional challenges for its detection (Apostolopoulos et al., 2021). Moreover, examining anti-debugging techniques reveals that attackers may engage in timing checks, use exception handling, or tamper with APIs, and the study suggests ways to manage and mitigate these tactics. Knowing about these evasion and obfuscation techniques allows us to make better detection systems that can manage and respond to modern threats.

2.2. Compiling from Source & Code Obfuscation


.netshrink v2.7 (Licensed to Bartosz Wójcik)

Input file: C:\MyApp\DllTestApp.exe

Output file: C:\MyApp\DllTestApp.exe

Bind virtual DLL libraries

Filename	File path
☑ MyFunctions.dll	C:\MyApp\MyFunctions.dll

Options

☑ Run monitoring thread to detect cracking and unpacking tools

☑ Password protection: reptilians are real ☐ Hide password

☐ Check .NET version: .NET v1.0 SP2, 32 bit

Select loader version: Automatically detect

Welcome to .netshrink

Figure 1 Screenshot of the .netshrink GUI showing options for compressing, encrypting, and protecting .NET executables, exemplifying code obfuscation and packing tools

By compiling malware, attackers are able to make adjustments to their software that help it avoid being spotted by antivirus software. If the source code is modified, the resulting effects are often unnoticeable, while the malware's goal does not change. Renaming variables, adding unnecessary steps, or including new functions that slow down the process or hide parts or errors from the tools used to analyze the program. Because of these points, it is hard for antivirus programs to identify the malware.

Code obfuscation is a useful method; it aims to make the code harder to understand and recognize. Altering the program's control flow, scrambling strings, or adding intrusive bits of code are examples of obfuscating techniques. If the authors' names and other, unique data are also removed, the effectiveness of these tactics is almost unbeatable.

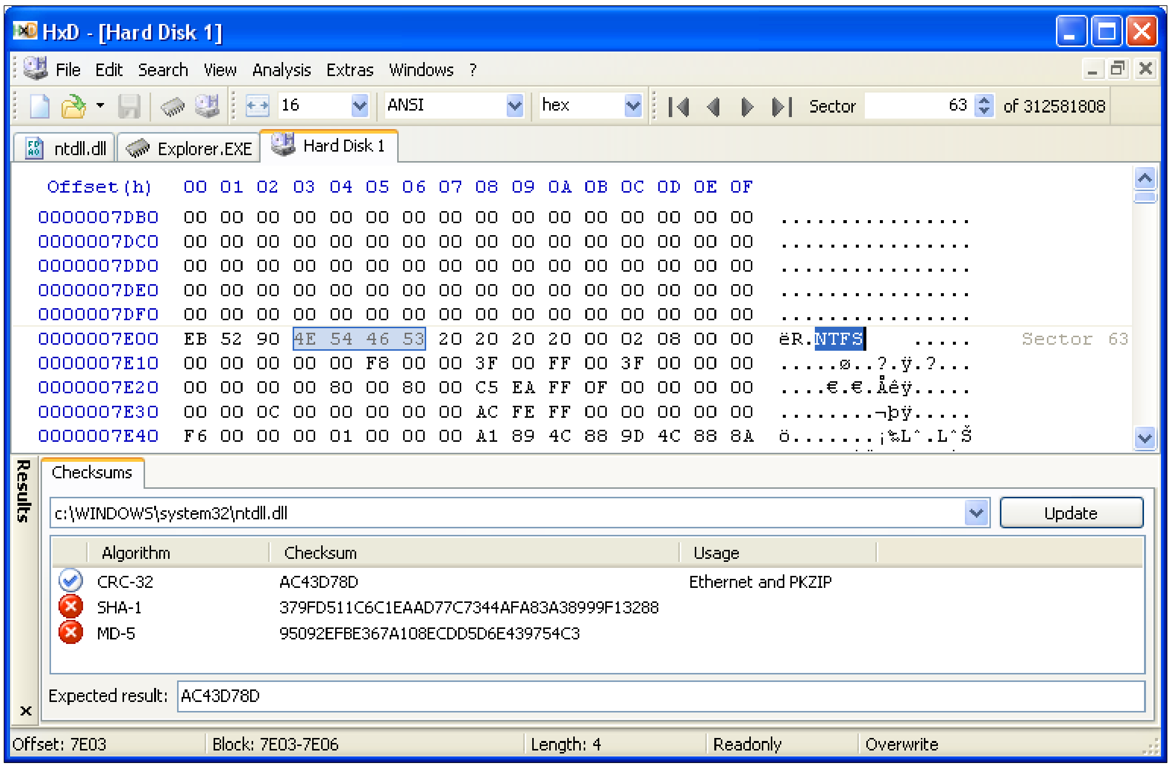
Tools like Visual Studio may be used to build and alter Quasar RAT, which is an open-source remote access trojan available everywhere. Attackers can purposefully modify the code of a malware before it is compiled to match and hide the malware in antivirus scans.

Extra protection for the compiled executable comes from .netshrink, in addition to what is provided by changing the source code. Image 4 outlines that .netshrink includes the ability to set a password, keep an eye on threads, and adjust settings for the .NET version and installer options. Because of these parts, it is more challenging to catch this malware with automated solutions.

By using obfuscation and .netshrink, the source code is compiled in a way that helps malware to stay hidden on the target computer for a longer period.

2.3. Hex Editing and Manual File Modification

Developers of malware and analysts manipulate the binary structure of files through hex editing. With such an approach, modifying and controlling the program's functions can be done using the executable without editing the source code it comes from. The code within the malware can be changed to hinder detection by removing giveaways and fixing spots that the antivirus program can identify.



```
 Offset(h)   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

0000007DB0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000007DC0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000007DD0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000007DE0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000007DF0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000007E00   EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00   ëR.NTFS    .....      Sector 63
0000007E10   00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00   .....ø..?.ÿ.?...
0000007E20   00 00 00 00 80 00 80 00 C5 EA FF 0F 00 00 00 00   ....€.€.Åêÿ.....
0000007E30   00 00 0C 00 00 00 00 00 AC FE FF 00 00 00 00 00   ........¬þÿ.....
0000007E40   F6 00 00 00 01 00 00 00 A1 89 4C 88 9D 4C 88 8A   ö.......¡‰Lˆ.LˆŠ
```

c:\WINDOWS\system32\ntdll.dll

	Algorithm	Checksum	Usage
✓	CRC-32	AC43D78D	Ethernet and PKZIP
✗	SHA-1	379FD511C6C1EAAD77C7344AFA83A38999F13288	
✗	MD-5	95092EFBE367A108ECDD5D6E439754C3	

Expected result: AC43D78D

Offset: 7E03 | Block: 7E03-7E06 | Length: 4 | Readonly | Overwrite

Figure 2 Hex Editor view of a Windows system file with checksum details showing CRC-32, SHA-1, and MD5 values, demonstrating binary-level file analysis and verification

Commonly, HxD and WinHex are used as hex editors for reading and changing the raw data in files or sectors. In Image 1, you can notice that HxD represents each value with hexadecimal and ASCII codes, which makes it simple to find and change specific information. Using this interface, you can view operations on the Windows system file ntdll.dll, and test their checksums against the CRC-32, SHA-1, and MD-5 algorithms. File integrity relies on checksums, and changing these values lets attackers functionally alter files without being detected.

It is possible to use hex editing to remove signs left by antivirus software within software files or to add code to evade detection by some antivirus tools. This practice allows people to manually alter the pattern of a virus to get past antivirus software. Statically detecting malicious files often fails as malware creators can change the files when being analyzed.

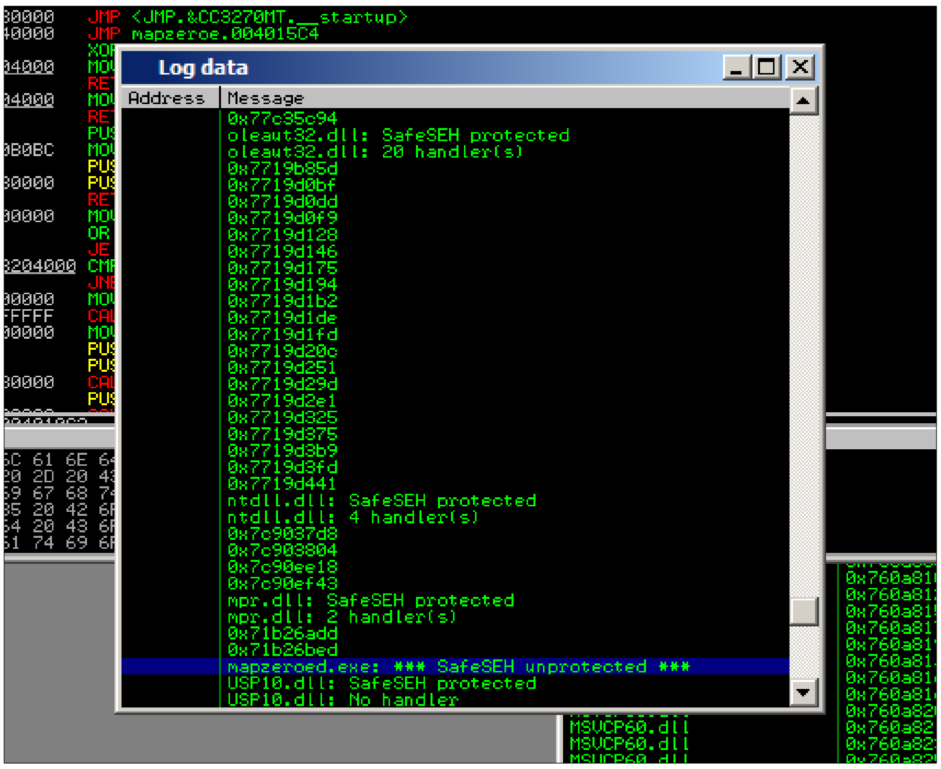
In addition, using hex editing is crucial for analysts investigating and analyzing malware. By looking at and analyzing binary content and comparing its checksums, analysts can uncover any changes or hidden threats. As a result, hex editors have a dual function in preventing and spotting viruses, making them crucial for modern malware research and cybersecurity efforts.

2.4. Debuggers and Anti-Debugging Techniques

Analysts use debuggers to investigate and observe the activities of malware. Yet, it's common for malware coders to add methods to detect and escape from being debugged. When malware detects that a program is being investigated, it may override its execution, close itself down early, or hide its true behavior.

Sometimes, the system checks the structures and handlers connected to debugging as an anti-debugging method. As an illustration, malware can examine the SafeSEH protections in the modules it loads to discover places that might allow debuggers to enter and hook into the program. As shown in Image 2, different system DLLs like oleaut32.dll, ntdll.dll, and mpr.dll show if they use SafeSEH or not, and give the number of exception handlers attached to them. If no SafeSEH was added, or patching was not done, to these modules as you noticed with ole32.dll and mspmsnsv.exe, this can mean that these files may be accessible for debugging and manipulation.

Such checks are used by malware so that it can notice if it is being fixed in a debugger and prevent its work from being interrupted. More anti-debugging methods involve timing delays, changing API calls, and taking advantage of any weaknesses in the debugger itself to either freeze or misdirect the analysis system.



```
JMP <JMP.&CC3270MT.__startup>
JMP mapzeroe.004015C4
```

```
0x77c35c94
oleaut32.dll: SafeSEH protected
oleaut32.dll: 20 handler(s)
0x7719b85d
0x7719d0bf
0x7719d0dd
0x7719d0f9
0x7719d128
0x7719d146
0x7719d175
0x7719d194
0x7719d1b2
0x7719d1de
0x7719d1fd
0x7719d20c
0x7719d251
0x7719d29d
0x7719d2e1
0x7719d325
0x7719d375
0x7719d3b9
0x7719d3fd
0x7719d441
ntdll.dll: SafeSEH protected
ntdll.dll: 4 handler(s)
0x7c9037d8
0x7c903804
0x7c90ee18
0x7c90ef43
mpr.dll: SafeSEH protected
mpr.dll: 2 handler(s)
0x71b26add
0x71b26bed
mapzeroed.exe: *** SafeSEH unprotected ***
USP10.dll: SafeSEH protected
USP10.dll: No handler
```

Figure 3 Debugger log data displaying Safe Structured Exception Handling (SafeSEH) protections across system DLLs, illustrating anti-debugging defenses and analysis indicators

Using SafeSEH and setting it up with proper configuration do make debugger-based attacks more challenging, but still, malware can overcome these protections. As a result, analysts need to use special debugging tools and methods to get around or handle security steps meant to stop them from analyzing the software.

Understanding how debuggers are stopped helps improve your skills in detecting and identifying malware. It demonstrates that malware keeps changing and that we must continue to find new ways to defend against it and solve forensic cases.

2.5. Packers, Cryptors, and Code Wrappers

Using packers, cryptors, and code wrappers, malware creators hide their harmful code, making it more difficult for antiviruses to detect. As a result, the altered executable looks different but works the same, blocking signature-based systems in popular antivirus tools.

Using packers, it is possible to compress or encrypt the content of executable files and put them into a secure container. When executed, the packer takes the code compressed or encrypted in memory and unpacks it so the malware can do what it wants to do. With packed files, antivirus software mainly analyzes the initial form of a file, not its actual code, so any investigation is made confusing. Some popular packers are UPX, ASPack, and VMProtect, and each one provides a different amount of protection from decompilation. Cryptors are designed in a similar way but rely on high-level algorithms to encrypt the payload, also often including anti-debugging and anti-emulation to make it more difficult to reverse engineer.

Code wrappers or joiners, on the other hand, put together different computer programs into one file, often mixing bad programs like viruses with good programs to help 'hide' the delivery. This combination of different malware hides the presence of the malware, making it less likely for it to be found during the first scan.

Recent studies and surveys have shown that modern antivirus programs are finding it hard to detect and stop runtime packers, since these types of malware can hide damage until they actually start running on a computer. These malicious packers often use API hooking, polymorphic code, and the creation of run-time code to prevent being detected more easily (Alkhateeb, Ghorbani, & Lashkari, 2023). When people behave this way, it allows malware to get past static scanners and sometimes beat dynamic analysis programs that use behavior to detect threats.

Moreover, figuring out and looking into hidden malware is still an area that many researchers are working on. Modern cryptors are designed to not only hide data by encryption, but also observe the environment to find virtual machines, sandboxes, and any debugging tools used. Because of these mechanisms, it can take longer for malware to be discovered in a system and it is able to damage these systems for a longer period (Alkhateeb, 2024). Consequently, security researchers keep coming up with new ways to break open and unravel packed or encrypted binaries, like using machine learning tools, in order to turn them back into simple code that can be studied and understood.

For this reason, it is necessary to learn about the actions and tactics of packers and cryptors to improve detecting malware. When antivirus software discovers common packaging and cryptographic styles, it can more effectively spot suspicious files in advance. Additionally, when you combine running checks of programs with looking at their signatures ahead of time, you get a stronger security system that can spot packed malware after it's been opened in memory.

In summary, packers, cryptors, and code wrappers are basic tools the creators of malware use to help their programs avoid being detected. Encrypting, compressing, and bundling by malicious developers makes it harder for traditional antivirus technologies to detect their malware. Advances in these approaches require continuous updates and efficient solutions to fight off obfuscated malware (Alkhateeb et al., 2023; Alkhateeb, 2024).

2.6. Loaders, Droppers, and File Binders

Many malware authors spread their software by embedding it in loaders, droppers, and file binders that are used to transfer it on target devices. Thus, by hiding the malware, segmenting it, or hiding it among several other files, the attackers manage to deliver it silently without raising suspicion.

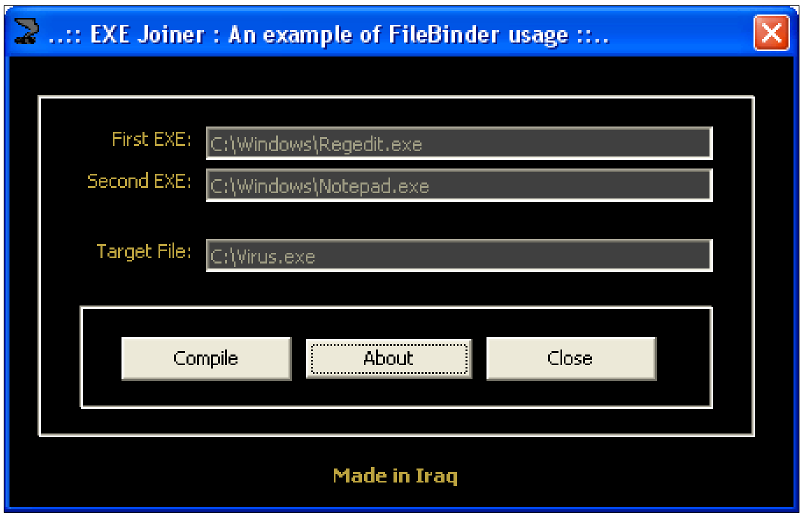
Loaders are small programs that help download and run the main part of the malware. They are often silent and only stay active for a little while, aiming to transmit the malicious part onto the machine they attack. Malware can be loaded onto a system by using scripts such as VBScript, JavaS, batch files, and PowerShell, which use permitted Windows functions to execute malware code. Droppers, on the other hand, are small files used to spread malware on a person's computer. The extra payload can be on the attack when downloaded or embedded in the files, and extra evasion techniques are often added to ensure things go undetected as installation happens.

These joiners put several files together to disguise dangerous code among the innocent files in a single archived file. This technique can be seen in Image 5, where an EXE Joiner tool puts together the two real programs (Regedit.exe and

Notepad.exe) into a new file called virus.exe. When malware is embedded in a secure program, there are fewer chances for detection and successful running of the code. Users are more likely to open programs they already know, which can accidentally start a malicious China DigiHunter virus.

Attackersoften use code obfuscation, crypting, and anti-debugging along with these methods to stay hidden and remain in the system for longer. Because droppers, loaders, and file binders can easily fool signature detection with their modified malware, they are still big hurdles for antivirus programs.

How infections spread is important to know in order to find the best ways to detect and control them. Looking at script actions, reviewing file information, and reviewing system behaviors, security tools protect the system better against the risks of loaders, droppers, and binders.


..:: EXE Joiner : An example of FileBinder usage ::..

First EXE: C:\Windows\Regedit.exe

Second EXE: C:\Windows\Notepad.exe

Target File: C:\Virus.exe

Figure 4 EXE Joiner tool interface combining multiple executables into a single file, illustrating file binding techniques used for malware delivery and evasion

2.7. Windows Security Mechanisms and Evasion

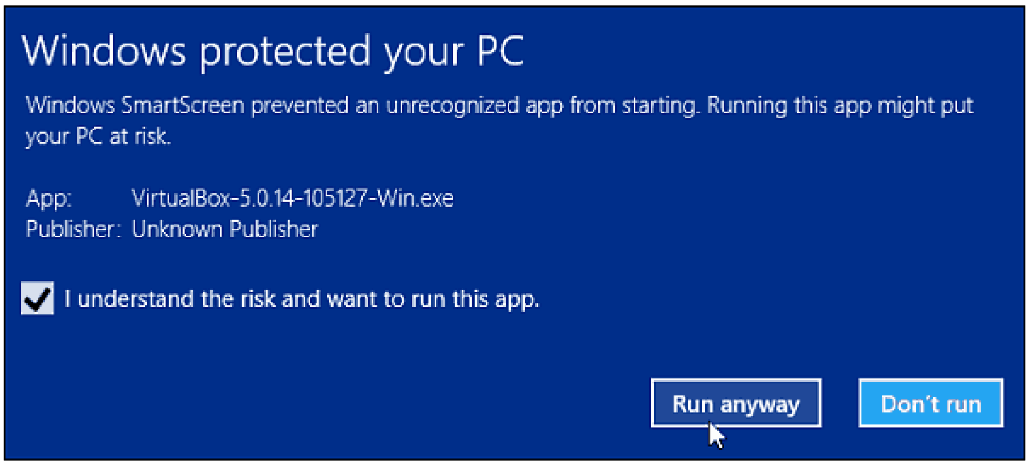
Windows operating systems include several security features that help protect users from getting infected by harmful software. A way Windows achieves this is through Windows SmartScreen, which filters out unknown or suspicious programs so they won't run until the user approves. It measures software using information about the publisher's background, the source of the file, and how common it is. When an application does not have a recognized digital signature or hasn't been used by many other people, SmartScreen will show you a warning, just like in Image 3. This dialog notifies users about the possible hazards and provides buttons for users to either block or permite the execution of the code.

Malware creators have come up with ways to get around or trick protection programs like SmartScreen and similar. With a fake digital certificate, malware can seem genuine and bypass the warnings given by SmartScreen. In some cases, attackers delay putting the program into action or rely on tricking people to carry out malicious applications despite any security alerts.

Malware authors may repackage or resign their software many times to trick systems into believing it is safe. Some types of malware try to exploit the update and approved list functions of SmartScreen in order to load themselves onto trusted paths.

While SmartScreen is designed to protect users, its dependance on what users do and the reputation of websites can lead to mistakes. Some users might forget to pay attention to the hints or not know about the risks, which enables the attackers. The human element and advanced evasion techniques give malware a way to get past security measures.

Being aware of the flaws and ways to carve around SmartScreen helps create strong defense plans. Using SmartScreen in combination with other similar systems can improve security and decrease the dependence on users' judgments.



Windows protected your PC

Windows SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

App: VirtualBox-5.0.14-105127-Win.exe
Publisher: Unknown Publisher

I understand the risk and want to run this app.

Figure 5 Windows SmartScreen security warning dialog preventing execution of an unrecognized app, highlighting user interaction in malware defense and bypass tactics

3. Methodology

3.1. Research Design

In this study, we use a qualitative approach to study how current malware evades antivirus products. By carefully looking at data, the research hopes to find out how malware authors can bypass detection. The method involves checking different evasion methods and types of malware to spot patterns and unusual ways attackers use them. Paying more attention to qualitative aspects than to numerical values allows the study to explain how these evasion techniques actually work and how well they do so. This setup makes it possible to look at a wide range of malware protection approaches and understand how threats evolve against current antivirus methods. It makes it possible to highlight both the benefits and drawbacks of various evasion techniques and to analyze the response of current endpoint protection products. All in all, this setup allows for a thorough investigation of how malware evasion methods are changing.

3.2. Data Collection

The data gathering step includes accessing both malware samples and evasion tools that users can find online. They use online repositories of malware and forums as well as open-source projects, where people share code and methods related to security. The malware samples included help display many different kinds and their methods of avoiding detection. Additionally, evasion toolkits and cryptors are used to find out what tools the attackers use to change or hide bad code. The data is gathered to have examples that are new and show what types of evasion methods are common right now. Implementing this technique helps to base the study on actual threats and keep up with the new ways malware authors exploit users. Gathering information from more than one platform allows the techniques to be checked against each other, increasing the reliability and scope of the study.

3.3. Case Studies/Examples

- Case Study 1: Ransomware Using Obfuscation and Packing Techniques Ransomware remains a common and troublesome type of malware that evades antivirus programs and wreaks the most damage it can. This study focuses on an advanced ransomware variant that included various methods to prevent detection for a long time and cause a lot of damage.

This attack relied mainly on obscuring the coding used in the ransomware. With this technique, the code was changed so that it became difficult for people and computers to detect malicious features, but it still carried out the same tasks. Ways of obfuscating the code were to rename things with meaningless names, introduce unnecessary steps, and play with the code's movement to confuse tools used by antivirus programs. Because obfuscation gave the ransomware new patterns, it avoided being detected by software that checked for well-known signatures of malware.

Further protecting itself, the ransomware was packaged with a runtime packer that made the executable smaller and unreadable. Executing the packer would make the malware's code difficult to see for analyzing tools searching the file, as the code would be unpacked in memory. The use of runtime unpacking creates more difficulties for antivirus systems, as they must use either simulation or other methods to detect suspicious things happening with the code.

Moreover, the malware was made from a modified version of the source code, which made it possible for attackers to include evasion methods that could avoid being spotted by specific antivirus products. With updated source code, anti-debugging tools that detect security analyst tools could now be added. If it finds an environment like this, the ransomware may halt its execution or end the process to circumvent analysis and delay being found.

The tool used by the ransomware was specially designed to carry its task safely and quietly. This small program was in charge of launching the unpacked malicious code, usually carrying out its task without showing any obvious signs. The loader separated the real malware code from the execution, hiding the malware and avoiding the risk of raising suspicion to detection tools.

An additional security measure used was file binding, where the ransomware combined with regular software was used. It involved bundling ransomware with common applications so it seemed safe at first sight. As soon as the users ran the tool, the hidden ransomware was set in motion. With this, the attackers easily installed the malware on devices because users trusted it and didn't doubt that it was suspicious.

As a result of these steps, the ransomware could remain in operation for a long while without being noticed, which allowed it to lock down critical information and force payments. Due to how the malware was well-obfuscated, packed, and made it tough for sandbox analysis, finding and solving the issue was put on hold.

This case points out how sleekly ransomware has become, and how hard it is for traditional antivirus programs to cope. It calls for implementing security strategies that use different types of analysis, such as signature-based, heuristic, behavioral, and runtime, to fight these types of threats. It also stresses the need for ongoing research on evasion methods and the creation of powerful countermeasures that spot and cancel out extremely hidden and packed malicious codes.

All in all, this case study proves that today's malware uses many creative methods to escape detection. Because ransomware threats are getting harder to detect, companies need to adapt by adding advanced detection, proactive threat collection, and educating staff to help fight them.

- Case Study 2: Banking Trojan Employing Anti-VM and SmartScreen Bypass

Targeting people's banking information, Banking Trojans are especially dangerous due to very sophisticated tricks to hide and avoid being noticed. In this case, a banking Trojan was able to steal confidential data by hiding in systems using a mix of anti-VM, anti-debugging, cryptors, dynamic unpacking, and SmartScreen bypass techniques.

A major way the Trojan evaded was by using anti-VM methods. It is common for security researchers to check malware in sandboxed or virtual machine environments to ensure their safety. But this banking Trojan had been programmed to spot such environments using analysis of different elements and time differences. When it detected being in a virtualized environment, the malware acted differently by either stopping or sending harmless responses, thus bypassing any analysis or detection in a sandbox. As a result, the malware could only run on true computers, making it more likely that searches would not find it.

As well as its anti-VM abilities, the Trojan also relied on anti-debugging features to deter analysis both by humans and automation. In other words, they reviewed whether debuggers were available, restructured how errors were handled, and ran tests to find if a slowdown was due to a debugger. In these situations, the malware responded with tricks such as closing down processes or messing up the testing, stopping any attempts to analyze it.

It used cryptors to hide the payload, thus dodging detection by antivirus programs based on virus signatures. Additionally, the unpacking process allowed the malware to decrypt and run its payload just when the system was being used, making it harder for detection tools to spot it. An antivirus running a static analysis on the file would likely miss a virus due to the application's ciphered state. After getting into the system, the malware used encryption to hide its payload, bypassing typical static and rule-based detection methods.

A dropper was used to ensure that the malware virus got onto the victim's computer. The malware came in the form of what appeared to be an everyday or trusted file, so it went undetected by many users.

The Trojan stood out for being able to bypass Windows SmartScreen, a function made to warn people before they run unidentified programs. The Trojan managed to pose as a reliable program by using stolen or fake digital certificates and relied on social engineering. As a result, users chose to ignore or disable the SmartScreen warnings, which let the Trojan bypass the built-in security.

Combining all the evasion strategies made it possible for the Trojan to get access to a system in a quiet manner and escape detection. Once onto the computer, it could look at what users do, gain access to their logins, and take out sensitive financial data, all without giving itself away.

It makes clear that banking malware has become sophisticated and that reliance on simple detection and reputation, like SmartScreen, may not be enough. It highlights the importance of connecting behavioral research, live system inspection, and user training to help spot and protect against such attacks.

To sum up, cybercriminals use anti-VM, anti-debugging, and cryptors, along with dynamic unpacking and Smart Screen bypass, to target financial information using the banking Trojan. By constantly updating security, it is possible to combat such threats and protect users from becoming victims of financial cybercrime.

3.4. Evaluation Metrics

The main part of evaluation is to find out how often these techniques work and how capable antivirus software is at finding and dealing with these threats. The success rate is the number of samples that pass by antivirus tools without being detected using tricks such as obfuscation or sandbox evasion. A higher number of successful evasions exposes issues with the current security systems. Antivirus detection rates measure how well a product spotts and stops dangerous software based on changing signatures, analysis, and how the malware works. You can see how effective endpoint protection is by comparing the detection rates of different antivirus companies. In addition, looking at how often TAs falsely identify normal files and frustrate users can be another insight. By merging all the metrics, we can see both the benefits and flaws of antivirus protection against evasive threats. To keep track of progress in cybersecurity and address any gaps, cyber defenses should be tested using real-life examples and modern evasion tools.

4. Analysis and Findings

4.1. Effectiveness of Code Obfuscation

Many antivirus tools find it hard to detect malware that is obfuscated. Making the code harder to read by obfuscating it helps it avoid signature-based detection, which depends on recognizing known chunks of code. It also makes it more difficult for researchers to analyze and understand the way malware functions, increasing the time and work they need to do. Methods of obfuscation like flattening control flow, inserting "junk" code, and encrypting strings make it hard for antivirus engines to properly detect threats. Even so, sometimes, trying to hide from detection can result in these steps adding more on top of the security while also making things buggy if proper care is lacking. The best antivirus solutions use heuristic and activity analysis to detect obfuscated malware, though these strategies are often outsmarted by new obfuscation strategies. All things considered, keeping code obfuscated is still a main way for malware to go undetected and remain active in target computers for a long time.

4.2. Impact of Hex Editing on Detection

Heavy use of hex editing helps malware writers affect programs in ways that antivirus scanners cannot detect using traditional signatures. Attackers can hide malware by modifying certain bytes to alter the known signatures or checksums, helping the malware pass as something harmless or original to antivirus software. Editing the manual file helps to get rid of noticeable traits that can be flagged by systems that rely on signature analysis. It is also possible to add code to a malware sample by changing the hex values, which can affect the way it behaves and makes analysis trickier. Even so, hex editing can be dangerous because making even the slightest error can stop the executable from working properly. Dynamic analysis in antivirus is now better at finding problems, regardless of some code modifications, but static signatures can still be easily manipulated. Hence, it is still important to use hex editing in developing malware, allowing it to stay hidden and avoid detection by early malware detection systems.

4.3. How Evasion Takes Advantage of Debugger Discovery

Malware writers use debugger detection to detect when someone is trying to analyze or break the code within it. When malware realizes a debugger is active, it tries to hide what it really does. This could cause the software to stop, generate wrong data, or cut off processing before it is done. As a result, it is harder for researchers to know the details of the malware and take actions to stop it. Ways to detect debuggers include looking at system processes, checking debug registers, and noticing delayed execution of code as a result of debugging. By staying unnoticed by debuggers, malware delays its detection and can complete its actions without interruptions. As methods of preventing debugging develop, tools for debugging follow suit by adding features to get past them. Learning about debugger detection in evasion plays a key role in improving ways to analyze and prevent malware.

4.4. Packers and File Binding Influence on Malware Stealth

Using both packers and file binding makes it harder for anti-malware tools to detect malware, as they change the coding and its form. Packing or encrypting malware software helps it disguise its contents so antivirus scanners won't recognize it. This runtime transformation covers up any patterns we might expect to find, so it makes it harder to use normal static analysis. Once packers are run, they unpack the payload in memory, which can often bypass software made to find unpacked static files. By combining malicious code with authorized software, file binding cover up malware to make it look safe. By using his approach, the attacker makes it easier for malware to run because people trust in these messages. Thanks to the efforts of packers and binders, detecting malicious components becomes more difficult, allowing malware to last longer in the target system. While dynamic analysis can uncover when malware is being unpacked by advanced antivirus, the number and, complexity of packed malware methods mean many of these samples still succeed in going unnoticed. Therefore, malware developers continue to rely on packers and file binding to help their malware avoid being noticed by security systems.

5. Discussion

5.1. Limitations of Current Antivirus Solutions

It is becoming increasingly challenging for current antivirus solutions to catch and block new types of malware. While signature-based detection blocks known viruses, it is unable to recognize variants of malware that keep changing their code. Though these methods do well in discovering new threats, they sometimes result in false positives, which make users uncomfortable and less confident in the system. Many anti-virus programs mainly use static scanning, which can still be avoided by using obfuscation during running or when files are packed. Malware using anti-debugging and anti-virtualization can dodge detection in regular dynamic analysis tools. Additionally, the limited resources on some devices can prevent scans from taking in much data or checking for many forms of threats. It is clear that malware is evolving faster than current antivirus can handle, and because of this, new more advanced, and flexible detection methods along with threat intelligence are required to fill existing gaps in current defenses.

5.2. Strengths and Weaknesses of Evasion Techniques

While these methods can help malware stay hidden, they bring certain weaknesses with them. They make it possible to hide malicious code by using obfuscation, encryption, and packing, making it hard for static detection to find the code. Anti-debugging and anti-virtualization is used to stop any in-depth analysis and allows malware to continue functioning without being noticed. At the same time, implementing these methods might increase malware complexity, which can affect how well it works or if it even runs correctly at times. In addition, new techniques can spot suspicious behaviors within the code, making evasion less likely successful. Updating the software and using machine learning help these solutions deal with newer attack techniques. Whilst evasion helps malware avoid being detected, it can still be spotted, so evaders have to regularly change their methods to keep ahead of antivirus advances.

5.3. Implications for Endpoint Protection Technologies

New ways for malware evasion are causing important changes in the field of endpoint protection technology. Traditional antivirus methods aren't enough anymore, so security companies are using different ways to protect computers by using a mix of methods like checking for known patterns, looking for suspect behavior, and using other methods to find and stop threats. These platforms need to include real-time monitoring, use controlled testing environments, and incorporate machine learning to catch and block tricky malware. Furthermore, solutions should adjust quickly to new threats by using up-to-date information about attacks and getting automatic updates. To deal with these new threats, companies need to check how the system works during use, instead of just analyzing static files. In the long run, these technologies should use better techniques to detect and block threatening actions, while maintaining performance and keeping users safe from sophisticated and sly cyber threats.

5.4. Recommendations for Malware Detection Improvements

It is important for security solutions to use a flexible and multi-layered approach to catch malware. When we apply artificial intelligence and machine learning, we can easily recognize malware that had not been seen before by examining its behavior, not just using a list of signatures. Sandboxes that carefully resemble actual user machines can spot malware that otherwise avoids being found by typical virtual machines. Improving active monitoring and analyzing memory will make it possible to sense when malware is being run in active processes. By sharing threat info, security vendors and organizations can more quickly detect fresh ways malware can avoid detection and various versions of malware. People should be informed to recognize risky behaviors and think twice before using apps they do not recognize. It is important to keep endpoint protection software up to date and patched, which helps defend against new and improved cyber threats.

5.5. Future Trends in Malware Evasion

Cybercriminals are expected to use improving technology to make their malware more difficult to discover. It is expected that attackers may adopt more AI to automate their ways of evading security measures and quickly adjust their malware behavior to best fit the system under attack. Given this, polymorphic and metamorphic malware will keep updating, causing almost one-of-a-kind malware in every attempt to infect. Malware that does not create any files on the hard drive will make it more difficult to detect attacks. They can also rely on trusted execution environments and cloud computing to cover any malicious activities they do. Using social engineering tactics, hackers will also likely attack human abilities and cause higher levels of confusion. As defenses get better, people who make malware will come up with new ways to do their job, which means they will try to be quieter, stick around longer on your device, and go after specific targets. It is important for cybersecurity to keep growing in order to predict and deal with these constant evasion schemes.

6. Conclusion

6.1. Summary of Key Points

The report points out that malware developers are finding new ways to avoid detection, which can greatly lower the efficiency of antivirus programs. Two of the main strategies are code obfuscation, packing, and crypting, all of which protect malicious code from signatures used in detection. Anti-debugging and anti-virtual machine tricks help malware escape being analyzed by detecting security tools and sandboxes. Malware can be secretly delivered on a victim's system with the use of loaders, droppers, and file binders, and Windows SmartScreen can be skirted through illegal ways. Making changes to binary patterns in files can make it more difficult for detection. Even with the improvements made in heuristic, behavioral, and machine learning-based virus detection, evasion methods are still a major cause of continued malware infections. All of these tactics work together to allow malware to get into computers silently, stay hidden, and do its destructive work. Since developers of malware are constantly getting better, it is important to have flexible and layered security systems. Understanding and dealing with these evasion tricks is important if we want to keep our online security better and help people avoid more complicated attacks.

6.2. Future Directions

In the future, as technology improves and attackers become more creative, the avoidance of malware by criminals will become an ongoing obstacle for security teams. One emerging trend is that hackers are increasingly using AI to make malware able to change its tactics and stay hidden. With both fileless malware and living-off-the-land techniques on the rise, the process of detecting attacks is going to become more complicated. The best way to fight against these threats is for security solutions to evolve with real-time analysis, use AI, and cooperate with others to identify and respond to emerging threats. Making the sandbox environments more advanced with user system similarities will help detect clever malware. Teamwork between researchers, industries, and government agencies will also make it easier to defend together. Staying up to date with new forms of cyber attacks calls for continuous research and new, flexible defense measures. Proactive strategies that include technology, help people understand how to stay safe, and good policies will be important for creating stronger cybersecurity systems that can deal with new malware tricks as they come up.

References

[1] Alenezi, M. N., Alabdulrazzaq, H., Alshaher, A. A., & Alkharang, M. M. (2020). Evolution of malware threats and techniques: A review. International Journal of Communication Networks and Information Security (IJCNIS), 12(3), December 2020.

[2] Alkhateeb, E., Ghorbani, A., & Arash Habibi Lashkari. (2023). A survey on run-time packers and mitigation techniques. International Journal of Information Security, 23(2), 887–913. https://doi.org/10.1007/s10207-023-00759-y

[3] Alkhateeb, E. (2024, August). Unmasking stealthy threats: Techniques for identifying and analyzing obfuscated malware. Lib.unb.ca; University of New Brunswick. https://unbscholar.lib.unb.ca/items/eac6c517-9310-4f5f-8a5e-7e0ff1454753

[4] Apostolopoulos, T., Katos, V., Choo, K.-K. R., & Patsakis, C. (2021). Resurrecting anti-virtualization and anti-debugging: Unhooking your hooks. Future Generation Computer Systems, 116, 393–405. https://doi.org/10.1016/j.future.2020.11.004

[5] C. Rohith and G. Kaur, "A Comprehensive Study on Malware Detection and Prevention Techniques used by Anti-Virus," 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2021, pp. 429-434, doi: 10.1109/ICIEM51511.2021.9445322.

[6] None Muhammad Saad, & None Muhammad Taseer. (2022). The Study of the Anti-Debugging Techniques and their Mitigations. International Journal for Electronic Crime Investigation, 6(3), 33–44. https://doi.org/10.54692/ijeci.2022.0603139

[7] Sahay, S. K., Sharma, A., & Rathore, H. (2019). Evolution of Malware and Its Detection Techniques. Information and Communication Technology for Sustainable Development, 139–150. https://doi.org/10.1007/978-981-13-7166-0_14

[8] Samociuk, D. (2023). Antivirus Evasion Methods in Modern Operating Systems. Applied Sciences, 13(8), 5083. https://doi.org/10.3390/app13085083