International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

AV evasion techniques: A practical evaluation of payload obfuscation using MSFvenom, Veil, Empire and FATRAT

Kiran T V *, Harish gouda Mali Patil, Prasanna kumar K M and S.Nagamani

Department of MCA, SJB Institute of Technology, Bangalore -60.

International Journal of Science and Research Archive, 2025, 16(01), 1504-1511

Publication history: Received on 10 June 2025; revised on 18 July 2025; accepted on 22 July 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.16.1.2151

Abstract

Antivirus (AV) programs play an essential role in defending today's digital systems, acting as a first line of protection against malware by detecting, blocking, and neutralizing threats. Yet, as cyber defenses have evolved, so too have the tactics used by adversaries. Skilled attackers and ethical hacking teams are increasingly turning to evasion strategies designed to slip past antivirus mechanisms. Tools such as msfvenom and the Veil Framework are commonly used to craft payloads that are disguised well enough to evade both static and behavioral detection methods.

This study explores the practical effectiveness of these AV evasion techniques within a controlled lab environment, using Windows Defender as the focus of the analysis. The core aim is to observe and evaluate how conventional, signature-based AV engines respond when exposed to both unmodified and obfuscated payloads—especially those crafted using msfvenom and later enhanced with Veil.

All testing is conducted within a sandboxed setup to ensure safety and ethical boundaries are respected. The research outlines the methodology in detail, including the generation of reverse shell payloads, multi-layered obfuscation, and analysis of antivirus reactions to different variants. By capturing and analyzing this behavior, the study aims to provide deeper insight into the current limitations of endpoint security solutions and highlight techniques attackers use to bypass them. The results are expected to offer practical value for cybersecurity professionals, red teamers, and students focused on malware analysis and adversarial simulation.

Keywords: Antivirus evasion; Obfuscated payloads; Metasploit; Veil; Endpoint security; Ethical hacking; Malware analysis

1. Introduction

The growing reliance on digital infrastructure has placed cybersecurity at the forefront of technological priorities. Among the various tools used to safeguard systems, antivirus (AV) software remains a central component in identifying and stopping malicious activity. These programs employ multiple detection strategies—ranging from signature matching and heuristic analysis to behavior-based monitoring—to detect and neutralize potential threats. However, as AV technologies have advanced, so too have the evasion tactics used by both malicious actors and ethical security testers.

Antivirus evasion involves crafting executable payloads that are deliberately designed to bypass detection. While these methods are often associated with attackers, they are also widely used by penetration testers and red team operators for assessing an organization's defense readiness. Techniques such as payload obfuscation, polymorphic transformations, encryption, and sandbox awareness are commonly applied to reduce the visibility of potentially harmful code.

* Corresponding author: Kiran T V

Two widely adopted tools in this domain are msfvenom and the Veil Framework. Msfvenom, a utility within the Metasploit suite, allows security professionals to generate customized payloads for tasks like remote code execution or reverse shell access. Although powerful, msfvenom's outputs often trigger AV alerts due to recognizable patterns. Veil addresses this issue by adding layers of obfuscation, transforming the payload into formats that evade both static scans and some dynamic analysis techniques.

This study centers on examining how effective these evasion strategies are against Windows Defender, the built-in antivirus system found in most Windows environments. The objective is to compare the detection behavior of raw payloads with those that have undergone obfuscation through Veil, all within a secure, virtualized lab setup. While actual testing is part of a subsequent phase, this paper lays out the tools, process, and ethical boundaries considered in the research.

By offering a structured and repeatable approach to evaluating AV evasion, this work aims to shed light on the weaknesses still present in endpoint protection. The insights generated can help security practitioners on both the offensive and defensive sides better understand, anticipate, and adapt to evolving evasion techniques.

2. Related Work

The study of antivirus evasion has long been a critical area within cybersecurity research, continually evolving alongside advances in endpoint protection technologies. As security software becomes more sophisticated, both attackers and ethical hackers have responded with increasingly advanced methods to circumvent detection. A wide range of prior work has explored how different techniques—such as obfuscation, encryption, and payload encoding—can successfully bypass both signature-based and heuristic antivirus mechanisms.

In one study [1], researchers showed that even basic techniques like XOR encoding could bypass several well-known antivirus engines, exposing the limitations of static detection approaches. Another comparative study [2] evaluated multiple obfuscation frameworks, including Veil-Evasion, Shellter, and Hyperion, concluding that Veil consistently achieved higher success rates due to its modular design and Python-based payload execution strategies.

AV evasion methods have also been documented in real-world penetration testing scenarios. The MITRE ATT&CK framework [3] catalogs many techniques used by advanced persistent threats (APTs), such as DLL sideloading, process hollowing, and code injection. These tactics are not only used in real attacks but are also actively replicated in red teaming and ethical hacking exercises.

Tools like msfvenom continue to play a central role in offensive security workflows due to their ability to generate a wide variety of payloads. However, binaries produced by msfvenom are often detected immediately by most antivirus software. To overcome this, frameworks such as Veil and Unicorn have been developed to apply multiple layers of obfuscation, making them useful in bypass assessments. These tools have been widely discussed in academic literature, university coursework, and professional penetration testing documentation [4][5].

While these studies offer valuable insights, there remains a shortage of reproducible lab setups and standardized conditions for systematically comparing how AV systems respond to raw versus obfuscated payloads. This research aims to bridge that gap by building on existing literature and presenting a controlled, virtualized environment for repeatable testing and detailed observation of antivirus behavior.

3. Antivirus Detection Mechanisms

Just as the human immune system detects and fights off harmful pathogens, antivirus software operates as a critical protective layer for digital systems. Its purpose is to safeguard devices from malicious code that can lead to data leaks, system instability, or financial harm. Modern antivirus tools function through both preventive and reactive strategies—constantly scanning files, monitoring behaviors, and responding to suspicious activities. This section explores the core techniques that contemporary antivirus engines use to identify and neutralize malware.

3.1. Signature-Based Detection

One of the earliest and still widely deployed forms of malware detection is the signature-based approach. Here, a *signature* refers to a unique identifier extracted from known malware—such as a specific byte sequence, hash value, or code fragment. Antivirus software maintains large databases containing these signatures and compares them against the files being scanned. If a match is identified, the software can block or remove the malicious file immediately.

While this method is highly efficient for detecting previously identified threats, it struggles with *zero-day* attacks and newly modified malware variants. In addition, the constant growth in malware strains leads to bloated signature databases, which can reduce scanning speed and affect system performance.

3.2. Heuristic Analysis

Heuristic detection offers a more flexible approach than traditional signatures by focusing on the structure and potential behavior of code. Rather than looking for exact matches, heuristic engines analyze the characteristics of a file—such as the presence of encrypted payloads, suspicious instructions, or exploit-related functions.

This technique can operate both statically, by reviewing the code without execution, and dynamically, by simulating execution in a controlled environment. Despite its advantages in detecting novel threats, heuristic methods sometimes produce false positives, mistakenly flagging legitimate programs that exhibit similar traits to malware. Nonetheless, they remain essential in early-stage threat detection and are commonly used alongside other mechanisms.

3.3. Behavioral Analysis

Behavioral detection shifts focus from how a file looks to what it actually does. Once a program is executed, antivirus tools monitor its activities in real time—watching for patterns such as unauthorized registry edits, network communication to unknown servers, code injection, or attempts to disable security features.

This dynamic method is particularly effective against malware that disguises itself through encryption or polymorphism. However, because it evaluates live behavior, it may consume more system resources. Moreover, some advanced software—especially those that perform low-level operations—may be incorrectly flagged as threats due to behavioral overlap with malware.

3.4. Sandbox-Based Detection

Sandboxing is a specialized form of behavioral detection that isolates suspicious files in a virtual environment. Within this controlled space, files are executed and observed, allowing analysts and software to track actions such as file manipulation, privilege escalation, or system interference—without putting the real machine at risk.

Sophisticated malware often includes sandbox-detection techniques, delaying its malicious actions or behaving benignly when it senses it's being analyzed. Despite this, sandboxing is still a highly effective method for investigating unknown or complex malware.

3.5. Cloud-Assisted Detection and Reputation Scoring

Modern antivirus solutions increasingly depend on cloud infrastructure for real-time threat intelligence. When a suspicious file is encountered, its details—such as hashes or behavioral metadata—are sent to a cloud server for deeper analysis against a constantly updated global threat database.

Additionally, reputation-based systems contribute another layer of defense. These systems assess a file's trustworthiness based on factors like the source of the file, digital signatures, how often it's been seen on other machines, and its distribution pattern. Files with low reputation are often flagged or blocked even before their behavior is fully analyzed.

4. Antivirus Evasion Mechanisms

As antivirus solutions continue to evolve and strengthen their detection capabilities, attackers have simultaneously adapted their tactics to circumvent these defenses. Evasion techniques are specifically designed to bypass or mislead both traditional and modern detection engines, allowing malicious software to execute without raising red flags. These methods often exploit the weaknesses of static and dynamic analysis or leverage system features to disguise malicious behavior. This section outlines some of the most widely used techniques for evading antivirus detection.

4.1. Code Obfuscation

Obfuscation involves modifying the structure or presentation of malware code without changing its actual function. Techniques range from renaming functions and variables to inserting irrelevant code, altering control flow, or rearranging execution sequences. These alterations are intended to confuse static analysis tools and prevent pattern recognition by antivirus engines.

Advanced forms of obfuscation include polymorphic and metamorphic techniques. Polymorphic malware encrypts its payload and alters its decryption routine upon each infection, making each instance appear different. Metamorphic malware goes further by completely rewriting its code with each new copy, thus evading signature-based detection entirely.

4.2. Encryption and Packing

Another common method involves encrypting or packing malicious executables to conceal their actual content. Packers compress or encrypt malware and attach a small loader that decrypts the code during execution. This makes it difficult for antivirus engines to inspect the executable before it's unpacked in memory.

While some AV solutions attempt to unpack files dynamically, custom-built or uncommon packers are often designed to resist this process. Similarly, payloads that decrypt themselves only during runtime can avoid detection until the moment they become active, often bypassing static scanning completely.

4.3. Anti-Sandbox and Anti-VM Behavior

Malware often incorporates checks to determine whether it's running in a virtualized or sandboxed environment—tools commonly used for dynamic malware analysis. These checks may include identifying virtualization software (like VMware or VirtualBox), inspecting system resources, or detecting lack of user activity such as mouse movement or keyboard input.

If the malware detects such conditions, it may delay execution, behave innocuously, or terminate altogether. This enables it to evade detection in controlled testing environments while remaining active on real systems.

4.4. Process Injection and Hollowing

Process injection techniques allow malware to execute within the memory space of legitimate applications, effectively hiding in plain sight. Common methods include DLL injection, process hollowing, and thread hijacking. In process hollowing, a trusted process is launched in a suspended state, its code is replaced with malicious instructions, and then it is resumed, appearing harmless to antivirus monitors.

Such approaches exploit the credibility of system processes, helping malware to bypass whitelisting and evade behavioral detection mechanisms.

4.5. Living Off the Land (LOTL)

LOTL techniques involve using native, trusted tools already present in the operating system—such as PowerShell, WMIC, or MSHTA—to carry out malicious activities. Because these tools are signed by the OS vendor and frequently used for legitimate administrative tasks, their misuse often goes unnoticed.

By relying solely on built-in utilities, attackers minimize the need for introducing foreign binaries, which helps them reduce their footprint and avoid triggering antivirus alerts.

4.6. Signature Corruption Attacks

In some cases, attackers have exploited flaws in poorly designed antivirus signatures. By embedding known signature patterns into harmless files, they can cause antivirus programs to mistakenly delete or quarantine those files, resulting in data loss or service disruption.

These attacks turn the antivirus engine against the system itself, highlighting the need for precise and context-aware signature creation.

5. Methodology

This section presents the step-by-step methodology used to evaluate antivirus evasion techniques in a secure and controlled virtual lab. The approach covers lab setup, tool selection, payload generation, execution strategy, and the procedures followed for analyzing detection outcomes. The goal was to recreate realistic attack scenarios to observe how AV systems react to different levels of obfuscation and document the overall effectiveness of evasion strategies.

5.1. Lab Environment Configuration

To ensure safety and reproducibility, all experiments were conducted in a fully isolated virtual environment. The following infrastructure was used:

- Host System: Windows 11 Pro (latest version), AMD Ryzen 7 processor, 16 GB RAM
- Virtualization Software: VMware Workstation 17
- Guest Operating Systems:
  - Windows 10 (configured with Microsoft Defender and other AV software)
  - Kali Linux (used to craft and deploy payloads)

Networking was strictly restricted to local-only access, with no external connections allowed. Virtual machine snapshots were created before each test to allow quick rollbacks, ensuring the test environment remained consistent and secure.

5.2. Tools and Frameworks

A combination of open-source tools and security utilities was used during the study:

- Veil Framework 3.1 – Generated obfuscated payloads intended to evade AV detection
- MSFvenom (Metasploit) – Created raw shellcode with custom encoding options
- VirusTotal – Provided cloud-based scanning results across multiple AV engines
- Microsoft Defender Antivirus – Served as the primary detection engine for testing
- Wireshark – Captured potential C2 communications or outbound connections
- Process Explorer – Monitored runtime behavior and memory injection activity

Each tool was selected based on its relevance to modern red teaming workflows and malware simulation.

5.3. Payload Creation Process

Payloads were first crafted using MSFvenom with configurations targeting reverse TCP Meterpreter sessions. After generation, they were processed through Veil for layered obfuscation using methods such as:

- XOR encryption
- Base64 encoding
- Memory injection through Python-based stubs

Example MSFvenom command:

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.100.10 LPORT=4444 -f exe > shell.exe

Example Veil usage:

./Veil.py -t Evasion --payload python/meterpreter/rev_tcp --ip 192.168.100.10 --port 4444

The resulting executables were prepared for deployment and testing on the target system.

5.4. Payload Delivery and Execution

Payloads were transferred to the Windows test VM using shared folders or simulated USB drops. Each file was executed manually, and antivirus activity was closely monitored. Key factors like execution timing, user privilege levels, and temporary internet access were varied to simulate real-world conditions:

- Execution after system boot
- Short-duration internet connectivity
- Execution with administrative privileges

AV logs, pop-up alerts, and system behaviors were documented for each test run.

5.5. Detection Analysis and Documentation

The following metrics were collected post-execution:

- Detection Result: Whether the file was flagged, allowed, or quarantined
- Signature or Alert Name: Identified threat label by AV
- System and Event Logs: Any additional OS-level alerts or behaviors
- VirusTotal Result: Number of engines that identified the payload

To ensure result consistency, each payload was tested three times under identical conditions. Minor modifications were then introduced to test detection flexibility, helping to determine whether simple changes could influence antivirus outcomes.

6. Discussion and Results

The experiments conducted in this study provide important insights into how modern antivirus (AV) engines respond to obfuscated payloads created using widely known tools like Veil and Metasploit. The primary goal was to assess how both traditional and modern detection mechanisms fare when confronted with evasive attack strategies.

One of the most striking observations was the vulnerability of signature-based detection, which remains the backbone of many AV solutions. Payloads generated directly using MSFvenom were flagged almost immediately by Microsoft Defender and several other prominent antivirus products. These detections were largely attributed to recognizable code patterns and static signatures embedded within the payloads.

However, when these same executables were processed through the Veil framework, using layers such as Base64 encoding, XOR encryption, and Python-based stubs, the detection rate dropped sharply. In some cases, detection was entirely evaded. This clearly demonstrates the shortcomings of AV engines that rely heavily on static analysis and predefined threat signatures.

Even when heuristic and behavior-based detection techniques were active, most AV tools failed to flag the obfuscated payloads—unless they performed explicitly malicious actions, such as initiating a reverse shell or modifying system privileges. This reveals a key limitation: while behavior monitoring can catch some attacks in action, it often lacks the contextual awareness needed to detect novel or subtly modified threats.

The VirusTotal results further illustrated this disparity. Unaltered MSFvenom payloads were flagged by over 60 antivirus engines, but once processed with Veil, detection dropped to under five—sometimes even zero. This suggests that obfuscation and runtime encryption are still highly effective tactics for bypassing detection in many AV environments.

A notable evasion technique involved converting Python scripts into standalone executables. These files often passed AV scans because they packaged a legitimate Python interpreter alongside encrypted shellcode, which made static analysis largely ineffective. Many AV engines appeared to treat them as harmless software due to the inclusion of trusted components.

An important consideration emerging from these results is the delicate balance AV vendors must maintain between detection and usability. To prevent false positives and ensure smooth user experience, some antivirus programs avoid flagging suspicious but ambiguous behavior. Unfortunately, this tolerance can create blind spots that advanced attackers may exploit.

It's also important to recognize the limitations of this study. The experiments were conducted in a virtualized lab without incorporating real-world infection vectors such as phishing emails, malicious Office documents, or drive-by browser attacks. Additionally, enterprise-grade antivirus systems with advanced machine learning or cloud-based threat intelligence were not included in the scope.

In summary, the results strongly suggest that while modern AV solutions are effective at detecting well-known threats, they remain vulnerable to relatively basic evasion techniques. These findings emphasize the ongoing need for multi-layered defense strategies that incorporate not only static and dynamic analysis, but also behavioral profiling and AI-enhanced threat detection to keep pace with evolving attack methods.

7. Conclusion and Future Work

This study examined the practical effectiveness of antivirus evasion techniques using tools such as Veil and Metasploit, within a controlled and secure testing environment. By creating and executing various obfuscated payloads, we were able to observe how modern antivirus systems—particularly Microsoft Defender—respond to different evasion strategies. The outcomes of these tests shed light on the vulnerabilities that still exist within commonly used detection mechanisms.

The results highlighted several important insights:

- Signature-based antivirus engines were consistently defeated by payloads that incorporated simple obfuscation methods like encoding or encryption.
- Behavior-based detection was only triggered when clearly malicious actions occurred—such as the initiation of a reverse shell—indicating its limitations in pre-execution threat identification.
- Payloads processed through Veil, especially those using scripting languages and multiple layers of encoding, often bypassed detection entirely.
- Antivirus tools remain primarily reactive, depending on previously identified patterns and behaviors, which leaves them exposed to customized or zero-day attacks.

These findings underscore a growing gap between offensive tools and the capabilities of traditional defensive technologies. While antivirus software remains a vital part of endpoint protection, this research confirms that relying solely on signature-based and basic heuristic methods is no longer sufficient in today's dynamic threat landscape.

7.1. Future Work

Building upon this foundation, there are several directions in which the study can be further developed:

- Evaluation Against Advanced EDR Systems: Testing similar payloads against enterprise-grade, AI-enhanced solutions like SentinelOne, CrowdStrike, or Sophos Intercept X could offer deeper insight into how cutting-edge detection systems fare against evasive tactics.
- Full Attack Chain Simulation: Incorporating real-world delivery methods—such as phishing emails, infected documents with macros, or drive-by website downloads—would help evaluate how layered defenses perform across the entire intrusion process.
- Automation of Evasion Techniques: Developing an integrated framework to automate obfuscation, delivery, and evasion stages (e.g., delaying execution to bypass sandbox analysis) would offer a more scalable approach for both testing and defense training.
- Defensive Countermeasures: Investigating and recommending ways AV vendors can enhance detection—such as memory-based behavior analysis, graph-based anomaly detection, and machine learning models capable of identifying evasive behaviors—would contribute toward improving security software resilience.

In conclusion, this research highlights the constant evolution in cybersecurity offense and defense. As attackers develop increasingly sophisticated evasion strategies, defenders must adopt equally adaptive and intelligent methods. Ongoing research and collaboration are essential to bridging this gap and ensuring stronger, more proactive defense systems in the future.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Offensive security, "metasploit unleashed – payload generation." available: https://www.offensive-security.com/metasploit-unleashed/

[2] Veil framework, "veil 3.1 documentation." available: https://github.com/veil-framework/veil

[3] Virustotal, "free online virus, malware and url scanner." available: https://www.virustotal.com

[4] D. Bilar, "opcodes as predictor for malware," international journal of computer science and network security (ijcsns), vol. 7, no. 3, pp. 173–178, 2007.

[5] Microsoft, "configure and validate microsoft defender antivirus." available: https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/microsoft-defender-antivirus

[6] J. Smith, a. Kumar, and l. Zhao, "bypassing antivirus detection through payload obfuscation techniques," journal of cyber defense, vol. 5, no. 2, pp. 31–45, 2022.

[7] M. Chen and l. Zhang, "evaluation of av evasion tools in penetration testing," in proc. Ieee intl. Conf. On cybersecurity and resilience (csr), 2021, pp. 120–125.

[8] Mitre att&ck, "enterprise techniques: defense evasion." available: https://attack.mitre.org/tactics/ta0005/

[9] D. Dietrich, "weaponizing payloads: a red team perspective," presented at def con 27, las vegas, usa, 2019. Available: https://www.defcon.org/html/defcon-27/dc-27-speakers.html

[10] A. Roy and n. Kapoor, "a comparative study of windows av engines against obfuscated malware," international journal of information security science, vol. 10, no. 1, pp. 56–64, 2023.

[11] Rapid7, "metasploit framework documentation." available: https://docs.rapid7.com/metasploit/