JSCE (Journal of System and Computer Engineering)
Vol.7, No.2, April 2026, pp. 158-165
ISSN (online): 2723-1240
DOI : https://doi.org/10.61628/jsce.v7i2.2623

Investigating LOLBAS-Based Malware Using Hybrid Analysis: A Case Study of PowerShell-Driven Fileless Execution

Rosmiati [1,a,*]; Muh. Ikhsan Amar [2,b]; Muhammad Arham Arsyad [3,c]; Hariani [4,d]
[1] *Sistem Informasi, Institut Teknologi Bacharuddin Jusuf Habibie, Jl. Balaikota No. 1, Parepare 91125, Indonesia*
[2] *Teknik Robotika dan Kecerdasan Buatan, Institut Teknologi Bacharuddin Jusuf Habibie, Jl. Balaikota No. 1, Parepare 91125, Indonesia*
[3] *Sistem Informasi, Institut Teknologi Bacharuddin Jusuf Habibie, Jl. Balaikota No. 1, Parepare 91125, Indonesia*
[4] *Teknik Informatika, Universitas Islam Negeri Alauddin Makassar, Jl. H.M Yasin Limpo No. 36, Gowa, 92118, Indonesia*
[a] *rosmiati@ith.ac.id;* [b] *ikhsan.amar93@ith.ac.id;* [c] *mrham1908@gmail.com;* [d] *hariani.kasim@uin-alauddin.ac.id*
** Corresponding author*

Abstract

This study aims to identify and understand the technical characteristics of the malware output.exe, obtained from the MalwareBazaar repository, through a hybrid reverse engineering approach. This method combines static and dynamic analyses to provide a comprehensive understanding of the malware's internal structure, execution behavior, and evasion techniques. Static analysis revealed the invocation of system functions such as CreateProcessW and RegSetValueExA, as well as the use of syscall to execute PowerShell commands directly, indicating the implementation of the LOLBAS (Living off the Land Binaries and Scripts) technique. Dynamic analysis using CAPE Sandbox confirmed the malware's actual behavior, including process injection into legitimate processes such as svchost.exe, launching powershell.exe for data compression, and establishing network communication via Discord Webhook for data exfiltration. Integration of both analyses shows that output.exe functions as an information stealer with fileless execution and advanced persistence mechanisms. These findings demonstrate that the hybrid analysis approach is effective in identifying modern malware that leverages legitimate system components to evade traditional signature-based detection methods.

Keywords *: malware analysis, hybrid reverse engineering, LOLBAS, PowerShell, fileless execution*

1. Introduction

Techological advancement have contributed significantly to society. However, they have also introduced various negative impacts(Fransisca & Ningsih, 2023; Nur Widiyasono et al., 2024). One of these impacts is the increasing prevalence of cybercrime, including the widespread distribution of malicious software (malware) (Li & Liu, 2021). Malware refers to harmfull software designed to exploit vulnerabilities in operating systems, application, websites, or network. It is considered a major threat in cybersecurity due to its ability to damage sytems or steal sensitive information(Aboaoja et al., 2022; Hidayat et al., 2025; Rosmiati et al., 2025). Malware is frequently used for information theft, primarily because of its capability to operate stealthily and evade direct detection by users(Aditya et al., 2024).

The evolution of malware has become increasingly rapid, with more sophisticated attack techniques emerging. One of the recent trends is the use of Living off the Land Binaries and Scripts (LOLBAS), a technique that exploits legitimate system tools to execute malicious

activities without requiring additional file downloads (Iavich et al., 2025). This approach makes malware significantly harder to detect, as its behavior closely resembles normal system processes.

Previous studies have explored malware analysis using static and dynamic approaches. Static analysis examines malware code without execution(Rosmiati et al., 2025; Widiyasono, 2025), while dynamic analysis observes malware behavior in a controlled environment (sandbox)(Damodaran et al., 2022). Each method has its own strengths and limitations, and when used independently, they often fail to provide a complete understanding.

Recent research suggests that hybrid analysis, which combines both static and dynamic approaches, offers a more comprehensive understanding of malware behavior and attack patterns (Gibert et al., 2022; Xu et al., 2022). However, studies focusing specifically on information-stealing malware that utilizes LOLBAS techniques remain limited. Therefore, further research is required to address this gap.

This study contributes to the existing body of knowledge by providing a focused investigation of LOLBAS-based malware, with particular emphasis on PowerShell abuse for fileless execution and data exfiltration. Unlike prior studies that primarily address general hybrid analysis framework, this work highlights the practical integration of static and dynamic analysis in uncovering real world infostealer behaviour that leverages legitimate system components for evasion.

2. Method

This study employs a hybrid analysis approach that integrates static and dynamic analysis to achieve a more comprehensive understanding of the technical characteristics of the malware. The research workflow is illustrated in Figure 1, which consists of three main phases: preliminary study, malware analysis, and reverse engineering.

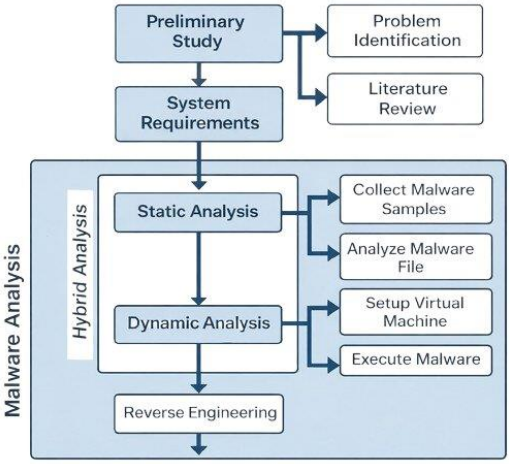

Figure 1. Research Stages

This phase begins with problem identification and a literature review. Problem identification is conducted to determine the key challenges in modern malware analysis, particularly those related to obfuscation and evasion techniques. The literature review examines previous studies focusing on malware analysis methods, both static and dynamic, in order to formulate the requirements for a relevant hybrid analysis system.

2.1 Malware Analysis

This stage represents the core of the study. In this phase, two types of analysis are conducted:

1. Static Analysis
   Static analysis is performed without executing the malware file(Adnyana, 2024; Jusoh et al., 2021; Rosmiati et al., 2025; Shaukat et al., 2023). This process includes collecting samples from the repository, identifying file structure using Detect It Easy and PE Studio, and decompiling the code using dnSpy and Ghidra to detect indications of packing, encryption, or code reflection.

2. Dynamic Analysis
   Dynamic analysis is conducted by executing the malware (Damodaran et al., 2022) using CAPE Sandbox and observing runtime activities such as network communication, file creation, registry modifications, and persistence behavior.

The results of both analyses are then integrated to provide a comprehensive understanding of the malware's structure and behavior.

2.2 Reverse Engineering

The final stage of the study is reverse engineering, in which the integrated results from static and dynamic analyses are used to reconstruct the malware's operational mechanism(Hazri, 2020; Muhammad Taseer Suleman, 2024). This process includes interpreting execution logs, analyzing code flow, and mapping the evasion techniques and data exfiltration methods employed by the malware

2.3 Research Data

The object of this study is a single malware sample used for technical analysis. The sample was obtained from the open repository MalwareBazaar, a platform that provides malware datasets for cybersecurity research and testing purposes. The malware analyzed is named output.exe, with the SHA-256 identification code: dc87dc8a4c8e5a3e841920e27ec046146b799facd8853f2dca767cf06f91499f.

SHA256 hash:	dc87dc8a4c8e5a3e841920e27ec046146b799facd8853f2dca767cf06f91499f
SHA3-384 hash:	237711d5c1626c5dd3e1ac0d1e9c3de743d217333aa6261393ad6e270254297abe4330c9b0be6fcd1efab953138d12a0
SHA1 hash:	c94742cc381520b47e0270fa819f684cc33e0307
MD5 hash:	ce492b36ca0957451877f3f3f5ac557f
humanhash:	papa-princess-jupiter-blue
File name:	output.exe
Download:	download sample
File size:	1'786'880 bytes
First seen:	2025-09-08 12:20:29 UTC
Last seen:	*Never*
File type:	exe
MIME type:	application/x-dosexec
imphash	1d8915c3554f512929a8d501df563d33 (1 x GenesisStealer)
ssdeep	49152:ql/ec5mVISRjU+lE2CtE6uRkKaADnT/OJUW:ql/b9tjaVDzOJU
TLSH	T1A3851226E7A514FDE16FC078CD124906EB72BC4A43A0E6DF27A04E665F23AE04D3D752

Figure 2. Malware Identification

This sample was randomly selected from a collection of malware categorized as an infostealer to ensure diversity in the techniques analyzed, particularly in the context of hybrid reverse engineering exploration. The malware file was downloaded with appropriate permissions for research purposes and subsequently stored and analyzed in an isolated laboratory environment. Prior to the analysis process, the file's authenticity was verified using its hash value to ensure data integrity and to avoid duplication with other samples.

3. Results and Discussion

3.1 Static Analysis

Static analysis was conducted on the output.exe malware sample to identify its initial characteristics and internal structure without directly executing the program. Based on the identification results using Detect It Easy (DIE) and PE Studio, the file exhibits a high entropy level, indicating the presence of packing techniques that obscure the original code structure.

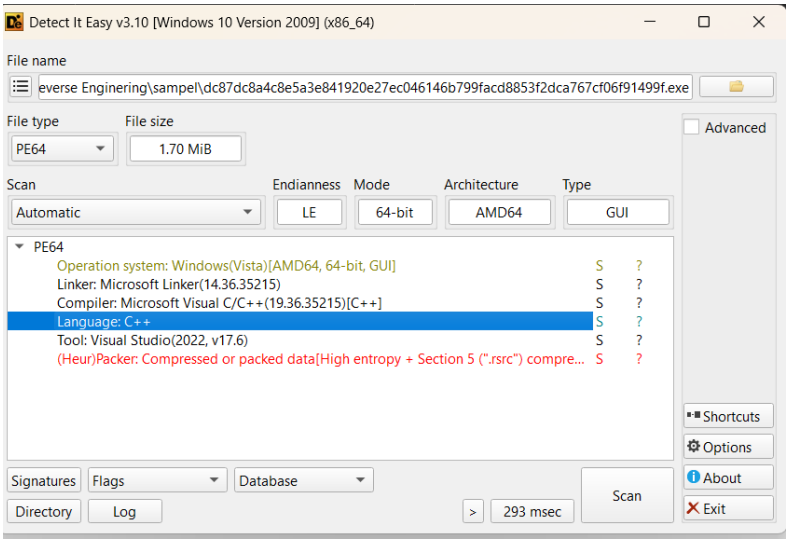

Figure 3. File Structure Based on Detect It Easy Analysis

In addition, metadata analysis indicates that the file was compiled using the C++ programming language, with several imported functions associated with system-level activities, such as CreateProcessW, GetAsyncKeyState, and RegSetValueExA. These functions suggest potential behaviors including input monitoring, registry modification, and the execution of new processes, which are commonly observed in information-stealing malware.

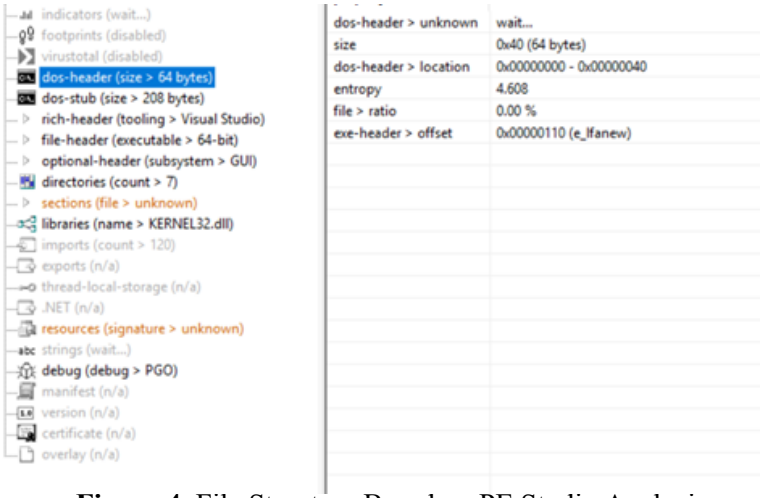

Figure 4. File Structure Based on PE Studio Analysis

The decompilation results using Ghidra reveal that output.exe contains several functional blocks related to process creation and memory manipulation. One significant

finding is the use of system calls (syscalls) to directly execute PowerShell instructions without relying on the conventional command prompt. This technique indicates the implementation of the LOLBAS approach, in which malware leverages built-in operating system components to perform malicious activities, making its behavior difficult to detect by traditional security software.

```
225       }
226       FUN_14003ab80((undefined (*) [32])(*pauVar8 + 0x18),pauVar8,lVar6 + 1);
227       FUN_14003ab80(pauVar8,(undefined (*) [32])"Terminating browser PID=",(ulonglong)pauVar11);
228       FUN_14003ab80((undefined (*) [32])(*pauVar11 + (longlong)*pauVar8),
229                     (undefined (*) [32])(pauVar11[0xa001f70] + 0x18),0x18 - (longlong)pauVar11);
230       pauVar8 = (undefined (*) [32])local_138;
231     }
232     _local_158 = *(undefined (*) [16])*pauVar8;
233     _local_148 = *(undefined (*) [16])(*pauVar8 + 0x10);
234     *(undefined8 *)(*pauVar8 + 0x10) = 0;
235     *(undefined8 *)(*pauVar8 + 0x18) = 0xf;
236     (*pauVar8)[0] = 0;
237     uVar16 = 0x14;
238     pauVar8 = FUN_14000d9a0((undefined (*) [32])local_158,(undefined (*) [32])" via direct syscall
        ."
```

Figure 5. Code Snippet from Ghidra Decompilation Showing PowerShell Invocation via Syscall

In addition, encrypted strings were identified that reference several PowerShell command-line arguments, indicating that the malware embeds Base64-encoded instructions within the main file. These instructions are subsequently decoded and executed directly in memory, supporting the hypothesis that output.exe implements fileless execution to evade file signature-based detection.

This sequence of activities suggests that the malware has a well-defined objective, namely to collect sensitive data from browsers, package it, and transmit it to an external attacker. The techniques employed, including target enumeration, process termination via syscalls, the use of PowerShell for data compression and exfiltration, and inter-process communication (IPC), provide strong evidence that this sample is a high-risk variant of information-stealing malware.

3.2 Dynamic Analysis

Dynamic analysis was conducted to observe the actual behavior of the malware during execution. The malware was executed in an isolated environment using CAPE Sandbox, with monitoring focused on process activity, system modifications, and network communication.

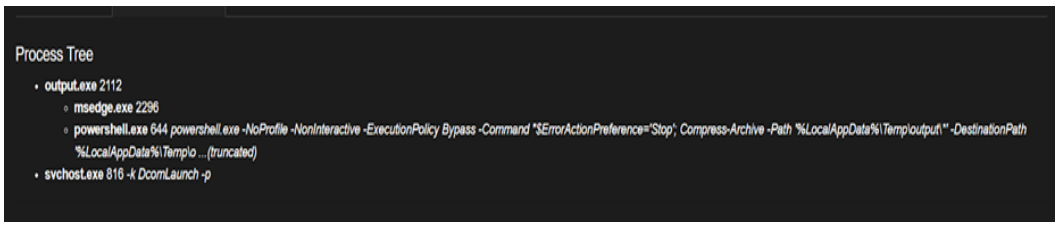

Figure 6. Process Tree from CAPE Sandbox Dynamic Analysis Showing Injection into svchost.exe

The observation results indicate that after execution, output.exe performs process injection into a legitimate system process, namely svchost.exe, in order to conceal its

malicious activities within a commonly used Windows process. This technique enables the malware to continue operating without raising suspicion from users or security systems.

Moreover, network communication activity was detected targeting a Discord webhook endpoint. Data transmitted over the HTTPS protocol includes browser credentials, cookies, and autofill data collected from browser storage directories such as Chrome and Edge. This activity further reinforces the identification of output.exe as a malware variant primarily functioning as an information stealer.

The behavioral report generated by CAPE Sandbox also shows that the malware creates multiple registry entries to maintain persistence after a system restart. As a result, even if the user terminates the active session, the malware can automatically execute again when the system is rebooted.

3.3 Hybrid Analysis Integration

The integration of static and dynamic analysis results provides a comprehensive understanding of the behavior of output.exe. Static analysis successfully reveals the internal structure and indications of PowerShell invocation, while dynamic analysis confirms the actual activities associated with these techniques during execution. The combination of these two approaches forms an effective hybrid analysis model, as it enables a comparison between the potential functionality inferred from the code and the actual behavior observed at runtime.

From this integration, it can be concluded that the malware implements three primary components: (1) process injection for concealment, (2) PowerShell-based LOLBAS techniques to execute malicious instructions without external files, and (3) HTTPS-based exfiltration to transmit stolen data to a command-and-control server. This hybrid approach enables more accurate detection of malware that employs fileless execution and LOLBAS, based evasion techniques.

3.4 Comparative Analysis of Static, Dynamic, and Hybrid Approaches

The effectiviness of each analysis approach can be assessed based on the type and depth of information obtained. Static analysis identifies structural indicators such as imported functions, encoded strings, and potential PowerShell usage; however, it is limited in confirming actual runtime behavior.

Dynamic analysis reveals execution patterns such as process injection, network communications, and persistence mechanism, but provides limited insights into the internal logic of the malware.

The hybrid approach integrates the strengths of both methods, enabling correlation between code-level indicators and observed runtime behavior. This integration imptoves the accuracy of malware characterization, particularly for detecting fileless and LOLBAS-based malware.

3.5 Limitations of the Study

This study has several limitations. First, the analysis is based on a single malware sample, which may not fully represent the diversity of LOLBAS-based threats. Second, the evaluation is primaly qualitative, focusing on behavioral interpretation rather than quantitative performance metrics.

Future work may address these limitations by analyzing multiple malware samples and incorporating quantitative evaluation methods to further validate the effectiveness of the hybrid analysis approach.

The findings of this study confirm that LOLBAS techniques and fileless execution introduce new challenges in cybersecurity, as most traditional antivirus detection mechanisms rely heavily on static file analysis. Detecting such activities requires a behavior-based approach, including behavioral analysis and correlation of system activity logs.

The hybrid approach employed in this study has proven effective in revealing the interaction between malicious code and operating system components. Practically, these findings can serve as a reference for developing advanced threat detection systems, particularly those based on PowerShell behavioral analysis and system call pattern monitoring.

4. Conclusions

This study demonstrates that the output.exe malware sample leverages the LOLBAS (Living off the Land Binaries and Scripts) technique through PowerShell to execute malicious commands without relying on external files. The integration of static and dynamic analyses confirms that the hybrid approach provides a comprehensive understanding of modern malware behavior, particularly those employing fileless execution and process injection techniques.

Furthermore, the comparative analysis highlights the advantages of the hybrid approach in correlating structural and behavioral indicators, thereby improving the accuracy of malware detection and characterization.

Despite its limitations, this study offers valuable practical insights into LOLBAS-based malware and contributes to the development of more adaptive and behavior-based detection strategies, especially those focusing on PowerShell activity monitoring and system event logging.

Acknowledgements

The authors would like to express their gratitude to the Directorate of Research and Community Service, Ministry of Higher Education, Science, and Technology (Kemdiktisaintek), for providing financial support through the 2025 Beginner Lecturer Research Grant scheme. The authors also acknowledge the support of LPPM-PM at Institut Teknologi Bacharuddin Jusuf Habibie (ITH) for their administrative assistance and guidance throughout the implementation of this research.

References

Aboaoja, F. A., Zainal, A., Ghaleb, F. A., Al-rimy, B. A. S., Eisa, T. A. E., & Elnour, A. A. H. (2022). Malware Detection Issues, Challenges, and Future Directions: A Survey. Applied Sciences, 12(17), 8482. https://doi.org/10.3390/app12178482

Aditya, H. N., Widiyasono, N., & Rahmatulloh, A. (2024). Analisis Malware Aquvaprn.exe Untuk Investigasi Sistem Operasi Dengan Metode Memory Forensics. Jurnal Teknik Informatika dan Sistem Informasi, 10(2), 161–172. https://doi.org/10.28932/jutisi.v10i2.6562

Adnyana, I. G. (2024). Reverse Engineering for Static Analysis of Android Malware in Instant Messaging Apps. Journal of Computer Networks, 6(3).

Damodaran, A., Troia, F. D., Corrado, V. A., Austin, T. H., & Stamp, M. (2022). A Comparison of Static, Dynamic, and Hybrid Analysis for Malware Detection. Journal of Computer

Virology and Hacking Techniques, 13(1), 1–12. https://doi.org/10.1007/s11416-015-0261-z

Fransisca, V., & Ningsih, W. (2023). The Advancement of Technology and its Impact on Social Life in Indonesia. Devotion : Journal of Research and Community Service, 4(3), 860–864. https://doi.org/10.36418/devotion.v4i3.445

Gibert, D., Planes, J., Mateu, C., & Le, Q. (2022). Fusing feature engineering and deep learning: A case study for malware classification. Expert Systems with Applications, 207, 117957. https://doi.org/10.1016/j.eswa.2022.117957

Hazri, M. (2020). Analisis Malware PlasmaRAT dengan Metode Reverse Engineering. Jurnal Rekayasa Teknologi Informasi (JURTI), 4(2), 192. https://doi.org/10.30872/jurti.v4i2.4131

Hidayat, M. R. T., Widiyasono, N., & Gunawan, R. (2025). OPTIMASI DETEKSI MALWARE PADA SIEM WAZUH MELALUI INTEGRASI CYBER THREAT INTELLIGENCE DENGAN MISP DAN DFIR-IRIS. Jurnal Informatika dan Teknik Elektro Terapan, 13(1). https://doi.org/10.23960/jitet.v13i1.5686

Iavich, M., Gnatyuk, S., Simonov, S., & Sydorenko, V. (2025). Taking LOLBAS Hacking to Another Level—Stealing Passwords using Built-in Binaries⋆. In Workshop on Cybersecurity Providing in Information and Telecommunication Systems.

Jusoh, R., Firdaus, A., Anwar, S., Osman, M. Z., Darmawan, M. F., & Ab Razak, M. F. (2021). Malware detection using static analysis in Android: A review of FeCO (features, classification, and obfuscation). PeerJ Computer Science, 7, e522. https://doi.org/10.7717/peerj-cs.522

Li, Y., & Liu, Q. (2021). A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. Energy Reports, 7, 8176–8186. https://doi.org/10.1016/j.egyr.2021.08.126

Muhammad Taseer Suleman. (2024). Malware Detection and Analysis Using Reverse Engineering. International Journal for Electronic Crime Investigation, 8(1), 109–123. https://doi.org/10.54692/ijeci.2024.0801191

Nur Widiyasono, Siti Rahayu Selamat, Angga Sinjaya, Rianto, Randi Rizal, & Mugi Praseptiawan. (2024). Investigation of Malware Redline Stealer Using Static and Dynamic Analysis Method Forensic. Journal of Advanced Research in Applied Sciences and Engineering Technology, 48(2), 49–62. https://doi.org/10.37934/araset.48.2.4962

Rosmiati, Amar, M. I., & Arsyad, M. A. (2025). Analisis Reverse Engineering Malware Winrar Sfx Menggunakan Ghidra Untuk Deteksi Teknik Obfuscation Dan Uac Bypass. Jurnal INSTEK (Informatika Sains Dan Teknologi), 10(2), 507–517. https://doi.org/10.24252/instek.v10i2.60503

Shaukat, K., Luo, S., & Varadharajan, V. (2023). A novel deep learning-based approach for malware detection. Engineering Applications of Artificial Intelligence, 122, 106030. https://doi.org/10.1016/j.engappai.2023.106030

Widiyasono, N. (2025). PENGANTAR ILMU ANALISA MALWARE.

Xu, P., Eckert, C., & Zarras, A. (2022). hybrid-Falcon: Hybrid Pattern Malware Detection and Categorization with Network Traffic and Program Code (arXiv:2112.10035). arXiv. https://doi.org/10.48550/arXiv.2112.10035