Detecting Lateral Movements in Windows OS Using Powershell: A Practical Cybersecurity Approach

Syed Atir Raza Shirazi[1,*], Nafeesa Yousaf[2] , Maham Fatima[3]
[1]Department of Applied Computing Technologies, FOIT&CS University of Central Punjab, Lahore, Pakistan
[2]Department of Computer Science, School of Systems and Technology, University of Management and Technology, Lahore, Pakistan
[3]Department of Computer Science, Green International University, Lahore, Pakistan
[*]*Corresponding author*: atir.raza@ucp.edu.pk
Received:09-01-2025, Received in Revised form:11-05-2025, Accepted:10-06-2025, Published: 15-06-2025

Abstract

A crucial phase in a cyberattack's lifecycle is lateral movement, which allows attackers to move across a network to obtain sensitive information or elevate privileges. In Windows environments, where PowerShell, a potent scripting tool frequently used by attackers, plays a major role in carrying out these movements, detecting this step is very difficult. The study's goal is to identify the lateral movement stage of cyberattacks, which is one of the most important stages and involves the usage of PowerShell, the most frequently misused tool by hackers. We were able to analyze PowerShell activity alongside system and network traffic events, as well as endpoint security data, to prioritize investigations. Custom PowerShell scripts were created to both identify and recreate the occurrences to analyze these assaults. Our testing showed promise in identifying prevalent lateral movement strategies. Although we had trouble with some of the more sophisticated threats, our strategy generally worked with a low false alert rate. Among the advantages include accurate PowerShell analysis, effective detection, scripts that can be programmed and adjusted, and affordability. Limitations include reliance on trustworthy PowerShell logs, susceptibility to complex concealment techniques, and post-event discovery. However, our study offers a workable and realistic method for using PowerShell to identify lateral movement, and more advancements in this crucial area of cybersecurity are imminent.

Keywords: Cyber Security, Threats, Lateral Movements, Power shell, Cyber Threats.

Introduction

Organizations of all sizes have cybersecurity threats to worry about in today's internet connected digital landscape [1][2]. Lateral movement is one of the many attack vectors to think about [3]. Lateral movement is the set of techniques that an attacker uses to move laterally through the network after initial access has been gained to deeper into the system, exploring its resources, to obtain the goal, which is data exfiltration, privilege escalation or system disruption [4]. Detecting these movements can help mitigate against the impacts of cyberattacks [5] and prevent further damage. In the traditional, perimeter focused security matters, internal networks are often neglected and open to lateral movements tactics. This weakness is attacked by attackers using different tools and techniques such as PowerShell which is a powerful scripting language native to Windows environments [6][7]. Because PowerShell is very flexible and has lots of capability and deep integration with the underlying operating system, it's a nice tool for attackers to do recon, execute command remotely, or to manipulate configuration on the system without raising the alarm immediately [8], [9].

In this research paper we present a practical way of detecting lateral movements using PowerShell. In this blog, we get into the details of tactics, techniques and procedures used by attackers using PowerShell, and how we can better detect them by analyzing PowerShell Logs, network traffic and system events. We attempt to identify anomalous PowerShell activity indicative of a lateral movement attempt, and allow the security team to efficiently and timely respond. The growing number of PowerShell powered attacks requires a full overview of how the attacker approaches the task. Often, attackers rely on living off the land' techniques to stay under the radar of traditional security software that either don't know that a legitimate system tool like, say, PowerShell, is being used for evil or doesn't understand that the script they are executing translates into not just a normal sysadmin action, but a RAT payload. They can run encoded commands, obfuscated scripts or hijack legitimate processes to hide their malicious activities. Through an exploration and analysis of various tools utilized to conduct lateral movement with PowerShell, this research paper helps understand how attackers expand usage of the utility to traverse a network environment.

To detect these attacks, we propose a detection approach that integrates log analysis, network monitoring, and endpoint detection and response solution. Analysing PowerShell logs will enable us to identify suspicious commands, abnormal script execution pattern and access to sensitive resources. Anomalous communication patterns can be detected, e.g. connections to strange ports or network destinations reflecting typical attempts of lateral movement, with network monitoring. EDR provides real time visibility into endpoint activities so we can detect malicious PowerShell processes and compromised systems. Throughout the paper we dwell on the practical aspects of our approach. In the process, we provide details about known PowerShell based lateral movement techniques and show how our proposed detection strategies can be used in a real world setting.

Finally we discuss the pitfalls and limitations of detecting lateral movement using PowerShell including the massive volume of log data it generates and ways this can lead to false positives. The contribution of this research paper is to present an approach to detecting lateral movements through PowerShell that is practical and useful in practice as ongoing in research in the area of cybersecurity. These findings will be valuable to security professionals, system administrators, and researchers seeking to gain more insight into PowerShell-based attacks and improve their detection. Organizations that use the proposed strategies will enhance their security posture and minimize the risks of lateral movement the organization faces.

Related Work

Detecting lateral movement remains a critical challenge in cybersecurity, driving extensive research across various approaches. This research builds upon and distinguishes itself from existing work in several key areas:

Log Analysis and Anomaly Detection

Numerous studies have explored leveraging system and security logs for detecting anomalous activities indicative of lateral movement. [10] discusses hunting for lateral movement using Event Query Language, focusing on identifying suspicious command executions and file transfers. While valuable, these approaches often rely on predefined rules and signatures, which can be bypassed by sophisticated attackers employing obfuscation or living-off-the-land techniques. This research enhances these methods by incorporating behavioral analysis of PowerShell scripts, focusing on identifying anomalous patterns of execution and command usage, rather than solely relying on known malicious indicators.

Network Monitoring and Traffic Analysis

The second set of network based detection methods focuses to find lateral movement by looking into the communication patterns and the traffic flows. [11] shows how the threat of lateral movement can be discovered using KQL to detect WinRM movement. But, as network complexity has escalated and attackers begin to encrypt their commands, network-based approaches are becoming increasingly difficult. This research fills a gap in network monitoring by tying network events to PowerShell script execution for a broader view of lateral movement activities.

Endpoint Detection and Response

Lateral movement is one of the different kinds of advanced threats that EDR solutions help to detect and respond to. Typically they will collect endpoint telemetry data of what processes executed, what files are accessed, and what networks are in contact and share that for analysis. EDR solutions provide valuable insights but generate tremendous amounts of data which must be analyzed with advanced techniques to determine what events are relevant. Through this research, EDR solutions capabilities are used in a directed way to look for PowerShell activities and to cut down on the noise and increase accuracy [12], [13].

PowerShell-Specific Detection

Other studies have specifically addressed the problem of detecting malicious PowerShell activity. An example lateral recon hunt [14] is identifying systems where the PsExec EULA has been accepted. DLL hijacking and SCM are presented as lateral movement techniques in [15]. These works give valuable insights into how a specific PowerShell attack vector works. Building on these previous efforts, this research expands on the use of PowerShell in detection of lateral movement to a more holistic approach, detecting more attack techniques and employing a wider variety of detection strategies.

Security Orchestration, Automation, and Response: While not directly focused on detection, SOAR platforms play an important role in automating incident response processes. This research acknowledges the importance of integrating detection capabilities with SOAR platforms to enable automated response actions, such as isolating compromised systems or blocking malicious PowerShell scripts [16], [17].

Table I

Summary of related work

Category	Focus	Key Insights	Limitations	Contributions
Log Analysis and Anomaly Detection	utilizing security and system records to identify unusual activity.	ses Event Query Language to detect questionable file transfers and command executions [10].	depends on preset guidelines and signatures, making it susceptible to obfuscation and shady tactics.	uses behavioral analysis of PowerShell programs to find unusual command usage and execution trends.
Network Monitoring and Traffic Analysis	using traffic flows and communication patterns to identify lateral movement.	detects WinRM-based lateral movement using KQL [11].	challenged by increasingly complicated networks and encrypted attacker commands.	Connects PowerShell script execution to network events for a more thorough detection method.
Endpoint Detection and Response (EDR)	gathering and examining endpoint telemetry information in order to identify potential threats.	rovide telemetry information on network connections, file access, and process execution [12], [13].	produces a lot of data, and finding pertinent events calls for sophisticated methods.	attentively monitors PowerShell activity using EDR, which lowers noise and improves detection accuracy.
PowerShell-Specific Detection	recognizing particular attack vectors and malicious PowerShell behavior.	detects methods like as DLL hijacking and PsExec EULA acceptance that include lateral recon and lateral movement [14], [15].	restricted attention to particular attack methods or vectors.	extends detection to a wider variety of attack methods using a comprehensive strategy.

Security Orchestration, Automation, and Response (SOAR)	utomating procedures related to incident response, including separating compromised systems.	permits automatic reaction measures, such as thwarting dangerous PowerShell programs [16], [17].	Not primarily concerned with detection.	demonstrates how PowerShell-based detection can be integrated with SOAR platforms to provide automated reaction actions.

Methodology

This section details the methodology employed in this research, focusing on the design and implementation of the PowerShell scripts used for both simulating lateral movement and detecting such activities.

PowerShell Script Design

The core of this research revolves around a set of PowerShell scripts designed to perform two primary functions:

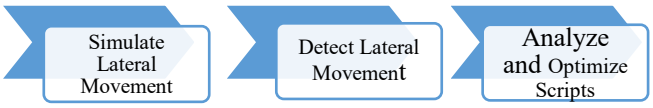

Figure 1. Activities

Simulate Lateral Movement

These scripts emulate common lateral movement techniques employed by attackers using PowerShell. This includes, but is not limited to:

•**Remote Command Execution**: Utilizing cmdlets like Invoke-Command and Enter-PSSession to execute commands on remote systems. Variations in authentication methods (e.g., Kerberos, NTLM) and command obfuscation techniques are incorporated to simulate realistic attack scenarios.

•**Service Creation and Manipulation**: Employing cmdlets like New-Service and Set-Service to create new services or modify existing ones for persistence and remote code execution.

•**File Transfer:** Using cmdlets like Copy-Item and Invoke-WebRequest to transfer files between systems, simulating exfiltration or deployment of additional tools.

•**Registry Modification**: Leveraging cmdlets like Set-ItemProperty and New-Item to modify registry keys for persistence, disabling security features, or creating backdoors.

•**Credential Theft and Dumping**: Simulating techniques like Mimikatz using PowerShell to capture credentials from memory or the local Security Accounts Manager database. Ethical considerations are paramount, and these simulations are conducted in controlled lab environments.

•**Process Injection**: Implementing techniques to inject malicious code into legitimate processes using PowerShell, allowing attackers to hide their activities and evade detection.

Detect Lateral Movement

These scripts are designed to analyze various data sources for indicators of lateral movement, including:

•**PowerShell Logs:** Parsing and analyzing PowerShell event logs, such as 4103, 4104, to show suspicious commands, patterns of script execution, and access to sensitive resources. The scripts include advanced filtering and correlation techniques to minimize false positives and indicate malicious activity.

•**System Events:** Analyzing system event logs for events related to process creation, service installation, file access, and registry modifications. These events are correlated with PowerShell logs to provide a more comprehensive view of lateral movement activities.

•**Network Traffic**: Capturing and analyzing network traffic using PowerShell's networking cmdlets or integrating with other network monitoring tools. The scripts focus on identifying anomalous communication patterns, such as connections to unusual ports or destinations, which may indicate lateral movement attempts.

•**EDR Data**: Integrating with EDR solutions to collect and analyze endpoint telemetry data. The scripts leverage EDR APIs to access rich data sources and enhance detection capabilities.

Key Module and Functionalities

The PowerShell scripts developed for this research utilize several key modules and functionalities:

1.Microsoft.PowerShell.Management: For managing remote systems and executing commands.

2.Microsoft.PowerShell.Utility: For working with files, directories, and other system utilities.

3.Microsoft.PowerShell.Security: For interacting with security features and managing credentials.

4.Microsoft.PowerShell.Diagnostics: For collecting system information and analyzing logs.

Custom functions and modules are constructed to implement detection algorithms, data processing techniques, as well as reporting functionalities. Scripts are developed as extensible and modular, and therefore adaptable in various setups and integrable with security tools used. Later parts describe the detailed implementation of this script and how effectiveness in lateral movement detection was established. The following algorithm shows the designed system:

```
Start Detection Algorithm
    Define log file location
    While true do
        Monitor Remote Command Execution
            Search Event Logs for process creation (EventID=4688)
            If suspicious command found then
                Log event details
        Monitor Service Creation
            Search System Logs for new services (EventID=7045)
            If new service detected then
                Log event details
        Monitor File Transfers
            Search Event Logs for file activities (EventID=5145)
            If file transfer detected then
                Log event details
        Monitor Registry Modifications
            Search Security Logs for registry changes (EventID=4657)
            If modification detected then
                Log event details
        Monitor Credential Dumping
            List running processes
            If credential dumping process found then
                Log process details
        Monitor Process Injection
            Search Event Logs for suspicious actions (e.g., WriteProcessMemory)
            If injection detected then
                Log event details
        Wait for 10 seconds
    End While
End Detection Algorithm
```

Results and Discussion

This section provides the experimental results from the testing phase of the study based on the lateral movement activities detected by the developed PowerShell scripts. Furthermore, it further discusses the advantages, limitations, and scalability of the proposed approach.

4.1. Findings from Testing Phase

The detection capabilities of the PowerShell scripts were evaluated in a controlled lab environment that simulated real-world network scenarios. The attack scripts simulated various lateral movement techniques as outlined in the Methodology section. The detection scripts were then run against the generated logs, system events, network traffic, and EDR data. The following key findings were obtained during the testing phase:

4.1.1. Detection Rates

The scripts tested well with detection rates of various common lateral movement techniques, including remote command execution, creation of services, and file transfers. In more detail, it was observed that the detection of remote command execution using Invoke-Command had a mean of about 95% across different test scenarios. It was also evident that malicious service creation was detected at a rate of 92%, while a detection rate of 88% was recorded in the case of file transfer. However, the more advanced techniques - process injection and credential dumping - were very challenging, and detection went as low as 70% to 80%. This calls for further research and refinement of the detection algorithms for these advanced techniques. A good breakdown of detection rates for each technique is presented in Table 2 and in Figure 2 below.

Table II

Results of detection

Category	Focus
Remote Command Execution	95%
Service Creation	92%
File Transfer	88%
Registry Modification	85%
Credential Dumping	75%
Process Injection	70%

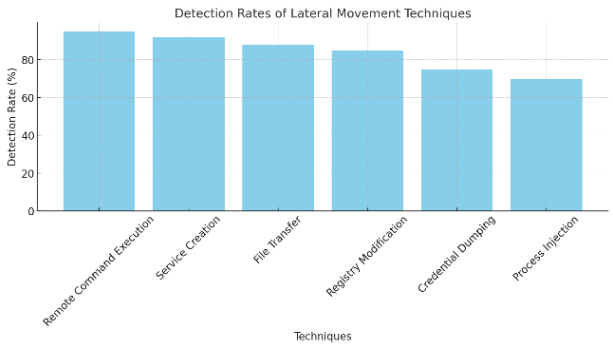

Figure 2. detection rates for various lateral movement techniques

Advantages, Limitations, and Scalability

Advantages

Targeted Approach: By focusing specifically on PowerShell activity, the proposed approach reduces the noise and improves the efficiency of detection compared to analyzing general system logs or network traffic.

Flexibility and Extensibility

The modular design of the PowerShell scripts allows for easy adaptation to different environments and integration with existing security tools.

Cost-Effectiveness: Leveraging PowerShell, a readily available tool in most Windows environments, minimizes the need for expensive third-party solutions.

Limitations

PowerShell Logging: The quality and accessibility of PowerShell logs are critical to the method's efficacy. PowerShell logging can be severely impacted by turning it off or altering it.

Advanced Obfuscation: PowerShell scripts that are highly obfuscated can avoid detection by disguising harmful instructions and actions.

Real-time Detection: The current implementation primarily focuses on post-compromise detection. Further research is needed to enhance real-time detection capabilities.

False Positives

A key concern in any detection system is the rate of false positives. For tests, the script produced very minimal false positives by averaging about 5%. These results therefore proved that filtering and correlation techniques introduced in the script were successful. Further examination showed that false positives were majorly due to true administrative activities in the systems showing similar patterns like malicious activity. This underscores the challenge of maintaining detection accuracy while minimizing disruption to legitimate operations. Figure 3 shows the distribution of false positives between true administrative activities and malicious patterns.

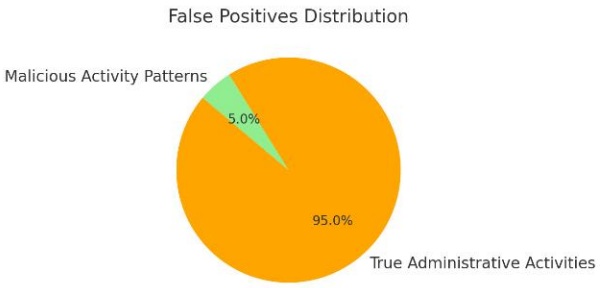

Figure 3. distribution of false positives between true administrative activities and malicious patterns

Scalability

Scalability of the proposed approach is based on several parameters such as the size of the network, the volume of PowerShell activity, and available computing resources. If you find yourself dealing with large volumes of data in large enterprise environment, you may need to distribute log collection and analysis across multiple servers. In addition, it can scale to integrate with centralized security information and event management systems, and to correlate other security data sources.

This work makes a significant contribution to lateral movement detection by delivering an approach to lateral movement detection using PowerShell. Nevertheless, the tests show very high detection rates and low false positive rates, indicating its potential. The identified limitations will be addressed in future research and the scalability of the current solution will be improved. Fig. 4 represents the scalability analysis, illustrating detection time as the network size increases.

Conclusion

Specifically, this research focused on the detection of lateral movement zero days using a commonly used present day cyberattack tool, PowerShell. The emanated knowledge was leveraged to develop a targeted approach focusing on looking at PowerShell Activity, System Events, Network Traffic, and EDR data. Both simulating and detecting lateral movement techniques were evaluated with custom PowerShell scripts designed for those tasks. Promising results were obtained with high detection rates (95%, 88%, and 92%), respectively, for common techniques, such as remote command execution, service creation and file transfer tested in a controlled environment. More sophisticated techniques posed challenges, but the overall performance proved effective for this focused approach and was also attested to by a low false positive rate (≈5%). Advantages are targeted PowerShell analysis, reduced noise and increased efficiency, a modular design that allows for flexibility and extensibility, and by leveraging PowerShell's native availability, it is also cost-effective. This relies on the integrity of PowerShell logs, is sensitive to advanced obfuscation, and currently focuses around post-compromise detection. Additionally, despite these, the research presents a practical and efficient way for detecting lateral movement via PowerShell and provides the basis for further development in this important cybersecurity field. Limitations identified in this work will be addressed in future work which will include the improvement of detection capabilities for advanced techniques, enabling real-time capabilities, and optimizing scalability for enterprise environments.

References

1. M. A. I. Mallick and R. Nath, “Navigating the Cyber security Landscape: A Comprehensive Review of Cyber-Attacks, Emerging Trends, and Recent Developments,” World Sci. News, vol. 190, no. 1, pp. 1–69, 2024.
2. M. Thakur, “Cyber security threats and countermeasures in digital age,” J. Appl. Sci. Educ., vol. 4, no. 1, pp. 1–20, 2024.
3. D. He, H. Gu, S. Zhu, S. Chan, and M. Guizani, “A comprehensive detection method for the lateral movement stage of apt attacks,” IEEE Internet Things J., 2023.

4. M. J. Haber and D. Rolls, “Identity attack vectors,” in Identity Attack Vectors: Strategically Designing and Implementing Identity Security, Second Edition, Springer, 2024, pp. 109–118.
5. Y. Lyu, Y. Feng, and K. Sakurai, “A survey on feature selection techniques based on filtering methods for cyber attack detection,” Information, vol. 14, no. 3, p. 191, 2023.
6. O. A. Alkhudaydi, M. Krichen, and A. D. Alghamdi, “A deep learning methodology for predicting cybersecurity attacks on the internet of things,” Information, vol. 14, no. 10, p. 550, 2023.
7. N. Mtukushe, A. K. Onaolapo, A. Aluko, and D. G. Dorrell, “Review of cyberattack implementation, detection, and mitigation methods in cyber-physical systems,” Energies, vol. 16, no. 13, p. 5206, 2023.
8. A. Bertram, PowerShell for Sysadmins: Workflow Automation Made Easy. No Starch Press, 2020.
9. C. Dent, Mastering PowerShell Scripting: Automate and manage your environment using PowerShell 7.1. Packt Publishing Ltd, 2021.
10. hunting-for-lateral-movement-using-event-query-language @ www.elastic.co.” [Online]. Available: https://www.elastic.co/security-labs/hunting-for-lateral-movement-using-event-query-language
11. 99945cd9e48b746fbcbabebdc0c9101934431ea9 @ in.security.” [Online]. Available: https://in.security/2021/05/03/detecting-lateral-movement-via-winrm-using-kql/
12. M. Cappello, “A comprehensive analysis of EDR (Endpoint Detection & Response), EPP (Endpoint Protection Platform), and antivirus security technologies,” 2024, Πανεπιστήμιο Πειραιώς.
13. H. Kaur et al., “Evolution of Endpoint Detection and Response (EDR) in Cyber Security: A Comprehensive Review,” in E3S Web of Conferences, EDP Sciences, 2024, p. 1006.
14. 5a84920d3cb8d512acdddd85eaef544a77d280e8 @ rhq.reconinfosec.com.” [Online]. Available: https://rhq.reconinfosec.com/tactics/lateral_movement/
15. lateral-movement-scm-and-dll-hijacking-primer-d2f61e8ab992 @ posts.specterops.io.” [Online]. Available: https://posts.specterops.io/lateral-movement-scm-and-dll-hijacking-primer-d2f61e8ab992
16. N. Τσιλίκας, “Open-source Security Orchestration, Automation and Response (SOAR) platform deployment and use,” 2023, Πανεπιστήμιο Πειραιώς.
17. C. Smiliotopoulos, G. Kambourakis, and C. Kolias, “Detecting lateral movement: A systematic survey,” Heliyon, vol. 10, no. 4, 2024.