machine learning & knowledge extraction

Article

Leveraging LLMs for Non-Security Experts in Threat Hunting: Detecting Living off the Land Techniques

Antreas Konstantinou [1], Dimitrios Kasimatis [1,*], William J. Buchanan [1], Sana Ullah Jan [1], Jawad Ahmad [2], Ilias Politis [3] and Nikolaos Pitropakis [1,4]

1 Blockpass ID Lab, Edinburgh Napier University, Edinburgh EH10 5DT, UK; 40479752@live.napier.ac.uk (A.K.); b.buchanan@napier.ac.uk (W.J.B.); s.jan@napier.ac.uk (S.U.J.); n.pitropakis@napier.ac.uk (N.P.)
2 Cybersecurity Center, Prince Mohammad Bin Fahd University, Al-Khobar 34754, Saudi Arabia; jahmad@pmu.edu.sa
3 Industrial Systems Institute, Research Center "ATHENA", Patras Science Park Building, Platani, 265 04 Patras, Greece; ilpolitis@isi.gr
4 Department of Information Technology, The American College of Greece, 153 42 Athens, Greece
* Correspondence: d.kasimatis@napier.ac.uk

Abstract: This paper explores the potential use of Large Language Models (LLMs), such as ChatGPT, Google Gemini, and Microsoft Copilot, in threat hunting, specifically focusing on Living off the Land (LotL) techniques. LotL methods allow threat actors to blend into regular network activity, which makes detection by automated security systems challenging. The study seeks to determine whether LLMs can reliably generate effective queries for security tools, enabling organisations with limited budgets and expertise to conduct threat hunting. A testing environment was created to simulate LotL techniques, and LLM-generated queries were used to identify malicious activity. The results demonstrate that LLMs do not consistently produce accurate or reliable queries for detecting these techniques, particularly for users with varying skill levels. However, while LLMs may not be suitable as standalone tools for threat hunting, they can still serve as supportive resources within a broader security strategy. These findings suggest that, although LLMs offer potential, they should not be relied upon for accurate results in threat detection and require further refinement to be effectively integrated into cybersecurity workflows.

Keywords: LLMs; artificial intelligence; threat hunting; security automation

check for updates

Academic Editor: Karin Verspoor

Received: 24 February 2025
Revised: 21 March 2025
Accepted: 26 March 2025
Published: 30 March 2025

Citation: Konstantinou, A.; Kasimatis, D.; Buchanan, W.J.; Jan, S.U.; Ahmad, J.; Politis, I.; Pitropakis, N. Leveraging LLMs for Non-Security Experts in Threat Hunting: Detecting Living off the Land Techniques. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 31. https://doi.org/10.3390/make7020031

1. Introduction

The rapid advancement and increasing prevalence of commercial Large Language Models (LLMs), such as ChatGPT, Gemini, and Copilot, have created new opportunities and challenges in cybersecurity. Threat actors are leveraging these models to enhance the efficiency and sophistication of their attacks despite existing safeguards designed to prevent misuse [1]. At the same time, defenders are exploring ways to utilise LLMs to stay ahead in an ever-evolving threat landscape. This is particularly relevant for smaller organisations that may not have the resources to maintain fully staffed, highly skilled security teams or invest in advanced security tools. For these organisations, LLMs present a potential means to augment their IT teams and carry out essential security tasks like threat hunting [2].

This paper investigates using LLMs, specifically ChatGPT, in the context of threat hunting for Living off the Land (LotL) techniques. LotL techniques, typically involved in Advanced Persistent Threat (AOT) attacks by malicious actors, use legitimate system tools and services to conduct malicious activities, which allow them to blend in with regular network traffic and evade detection by conventional security measures [3]. Cloud computing integration has further exacerbated these types of malicious insider attacks [4] by providing an increased cyber threat landscape. Such an activity often appears benign to automated tools, making it challenging to identify using standard detection methods. Detecting these techniques is challenging because attackers leverage existing administrative tools and processes routinely permitted in enterprise environments. Specifically, utilities such as PowerShell scripts, Windows Management Instrumentation (WMI) queries, and Remote Desktop Protocol (RDP) sessions are integral to regular IT operations for tasks like remote management, system diagnostics, and automation of administrative workflows. APT actors can exploit these functions by executing encoded or obfuscated PowerShell commands, crafting malicious WMI event subscriptions to achieve persistence, or initiating unauthorised lateral movements by leveraging legitimate RDP sessions [5]. Consequently, traditional detection systems, especially those based on signature matching or static blacklisting of malicious files and packets, frequently miss the intrusions that adopt the guise of legitimate traffic. Threat hunting is a proactive defence strategy that involves searching for signs of potential compromise, and it often focuses on subtle indicators that automated systems might miss [6]. This research seeks to determine whether LLMs can generate accurate and effective queries across a range of security tools to aid in identifying LotL techniques. The goal is to assess the potential of LLMs to enhance the capabilities of IT teams, especially those lacking specialised threat hunting expertise.

While much of the current research on LLMs in cybersecurity focuses on their use in areas like incident response, detection engineering, and process development, there is a noticeable gap in the literature regarding their application in proactive defence measures like threat hunting. This is particularly important for detecting LotL techniques, which allow threat actors to operate undetected by automated tools. Threat hunting is a specialised discipline that requires a deep understanding of system behaviour and threat actor methodologies. Consequently, many smaller organisations, which often lack the resources or expertise for dedicated threat hunting activities, find implementing effective proactive defence strategies challenging [7].

The absence of regular threat hunting is common across organisations of all sizes and in various sectors, including those adopting Industry 4.0 practices and deploying Internet of Things (IoT) devices. These technologies can significantly expand the attack surface, adding layers of complexity to an already challenging cybersecurity environment [8]. Often, the barriers to proactive defence are not purely technical but also stem from a lack of confidence and skill among security personnel. One example involves a supplier account compromise attributed to the threat group Wizard Spider [9]. By exploiting native system tools, the attacker moved laterally through the network with escalated privileges and was only detected by Endpoint Detection and Response (EDR) when using Cobalt Strike during post-exploitation. Until then, their use of legitimate binaries allowed them to evade automated detection. In contrast, successful threat hunting shows the value of proactive measures: in its report [10], IBM showcases that the average lifespan of an undetected threat in a system is 194 days, which can equate to millions in damages for businesses. As organisations integrate IoT and Industry 4.0 technologies, the stakes for establishing regular threat hunting practices become even higher, underlining the need for specialised expertise and the right resources to protect increasingly complex infrastructure.

This paper explores whether LLMs can effectively generate queries for security tools, assisting defenders, particularly those without specialised cybersecurity expertise, in proactively identifying malicious activities such as LotL techniques. The research anticipates that LLM-generated queries will reliably improve threat hunting capabilities, enhance detection accuracy, and lower barriers to effective cybersecurity for organisations with limited resources. By examining these capabilities, the paper seeks to clarify the role of

LLMs in strengthening security practices and promoting a more proactive cybersecurity posture. Specifically, the key objectives of the research are as follows:

- Evaluate the reliability of LLM-generated queries in accurately detecting malicious activity within common security tools without manual intervention.
- Assess how prompt complexity, i.e., simple to prescriptive, impacts the quality, accuracy, and practical usability of the queries generated by LLMs.
- Investigate the extent to which non-security professionals can effectively use LLM-generated queries to detect LotL techniques, thereby reducing dependence on advanced cybersecurity knowledge.

2. Background Knowledge

2.1. Threat Hunting

Threat hunting is a proactive security practice that goes beyond reacting to known threats. Instead of relying solely on automated systems, skilled professionals actively search for subtle signs of malicious activity that may not trigger conventional alarms. They draw on threat intelligence and knowledge of attacker tactics, techniques, and procedures (TTPs) to identify unusual patterns and behaviour [11]. This approach can reduce the amount of time attackers remain undetected inside a network, improving the effectiveness of incident response. It also encourages continuous improvement, as insights gained from hunting activities can point to weaknesses in monitoring, logging, and other controls.

Despite these advantages, threat hunting takes work. It often involves sifting through large volumes of data in complex environments, a task that requires highly trained professionals [12]. The cybersecurity industry faces a serious skills shortage, with millions of unfilled positions, making it hard for many organisations to attract or develop the required talent [13]. Threat hunters need a strong understanding of digital forensics, incident response, and network and endpoint behaviour. Developing these skills can be costly, which is a significant barrier for smaller organisations.

Following a structured threat hunting process, such as the one displayed in Figure 1, can help. Organisations often start with a clear aim and form hypotheses based on current threat intelligence [14]. They then select data sources and analysis methods to test these ideas. Findings from this process may guide incident response if threats are found or lead to improvements in security controls and detection capabilities. Over time, this cycle supports a more informed and adaptive defence strategy.

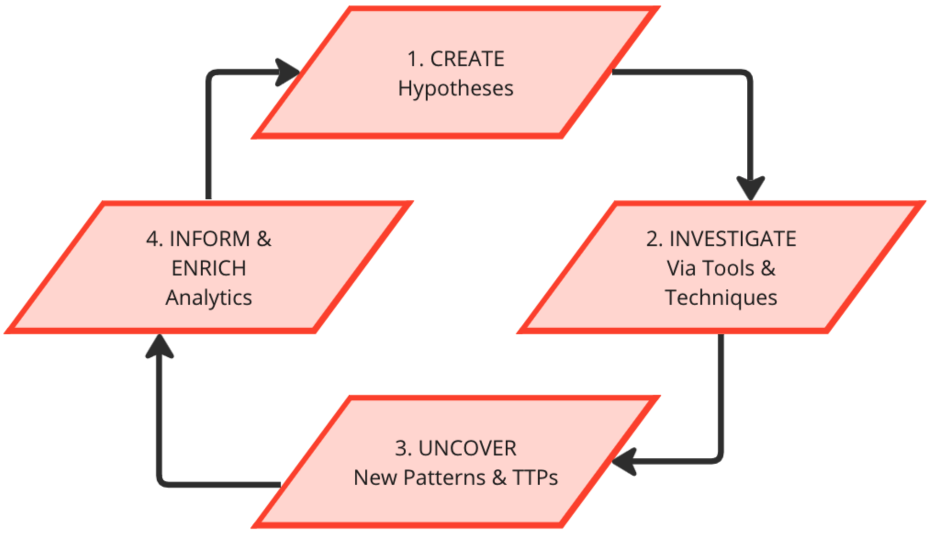


Figure 1. Threat hunting lifecycle, including attacker tactics, techniques, and procedures (TTPs).

2.2. Living off the Land Techniques

As attackers grow more skilled, they increasingly rely on methods that do not fit the traditional profile of malware. One such approach is the use of LotL techniques, where threat actors exploit trusted, built-in system tools instead of deploying harmful code that security tools may easily recognise [3]. By abusing native administration utilities, attackers blend their actions into everyday network activity, making it much harder for defenders to spot them.

Common tools used in LotL attacks include PowerShell and Bash scripting environments, which allow attackers to automate tasks, carry out system reconnaissance, execute arbitrary code, and move laterally within a network [14]. Threat actors also misuse features like Windows Management Instrumentation (WMI) for remote command execution and data exfiltration. Even basic administrative utilities, such as PsExec, net, or schtasks, can be turned against the organisation to gain persistence, create backdoors, or change system settings without arousing suspicion.

By using legitimate, signed binaries that are already present in the environment, LotL attackers can bypass traditional signature-based detection methods and application allowlisting [15]. This stealthier approach means that common warning signs of compromise are often missing. As a result, defenders cannot rely solely on known Indicators of Compromise (IoCs) and must look more closely at normal activity to detect when something is amiss [16]. There are still measures that organisations can take to counteract these threats. Shifting focus from signature-based detection to behavioural analysis helps identify unusual patterns, revealing hidden LotL activity. Strengthening applications allows listing and restricting user privileges, which can further limit what an attacker can achieve if they gain a foothold. Ultimately, proactive threat hunting that looks specifically for LotL techniques can catch these threats before they cause serious damage, but it requires skilled analysts who understand how legitimate tools are normally used and can detect subtle deviations.

2.3. Large Language Models

LLMs represent a significant development in natural language processing. Built on the Transformer architecture, these models use self-attention to learn how words in a sentence relate to each other, rather than processing text in a fixed sequence [17]. LLMs can understand context and meaning by analysing large volumes of unstructured text, enabling tasks such as text generation, translation, and summarisation [18].

Training LLMs generally happens in two main stages. First, a pre-training phase exposes the model to vast amounts of unlabelled text, teaching it to identify language patterns and structures. This is followed by fine-tuning, where a smaller, task-specific dataset is used to guide the model towards more focused objectives [18]. The result is a flexible model that can support many language-related tasks, reducing the need to build separate models from scratch for each application [17].

Still, LLMs face several limitations. They can produce information that sounds plausible but is factually incorrect, reducing their reliability in scenarios where accuracy is critical [19]. They can also inherit biases present in their training data, potentially reinforcing stereotypes and unfair assumptions [20]. Moreover, LLMs lack true understanding and can struggle with reasoning, abstract thinking, or interpreting figurative language [21]. These concerns raise important ethical questions. LLMs can generate harmful content, mislead users, or produce outputs that influence public opinion [22]. Biases learned from training data can lead to discriminatory outcomes, affecting decision-making processes in sensitive areas. Furthermore, the complexity and opacity of LLMs make it challenging to explain how they reach their conclusions, undermining trust and accountability [23].

To address these issues, developers and researchers are exploring methods to reduce biases, improve transparency, and increase accountability. Approaches like explainable AI seek to clarify how LLMs produce their outputs, while governance frameworks and industry standards help ensure responsible development and deployment [20].

2.4. AI in Security Tooling

Commercially available AI-driven cybersecurity tools claim to detect and respond to threats by applying advanced models, including LLMs, to network data. Services like Microsoft's Copilot [24] and CrowdStrike's Charlotte AI [25] promise to assist teams with everyday security tasks, incident response, and threat management. While these tools may offer value, they often come at a high cost and require a certain level of security maturity and expertise to use effectively. Smaller organisations and teams without specialist skills may not benefit as much since powerful features can go underused if staff are not trained to leverage them fully.

2.5. Prompt Engineering

One key practice when working with LLMs is prompt engineering: formulating input instructions to guide the model towards useful outputs [25,26]. Clear and specific prompts help ensure that the model produces relevant results. For example, when using LLMs to support threat hunting, well-structured prompts might generate queries that highlight unusual network behaviour or surface suspicious patterns in logs.

Effective prompt engineering involves careful planning. It is important to define the goal, provide the proper context, and use examples that show the desired output. Aligning prompts with data that the model can understand, iteratively refining questions, and adjusting language to suit the model's capabilities can all improve the quality of responses [27].

However, prompt engineering also has its challenges. Biases in training data can influence outputs, and small changes in prompts can produce unpredictable results [28]. There is no standard approach, and each LLM may behave differently. Finding the best prompt can be time-consuming, and assessing the quality and accuracy of outputs is not always straightforward.

In threat hunting, prompt engineering might involve providing threat intelligence as context, specifying the type of malicious activity to look for, and choosing the right query language for security tools like Splunk [29]. The process is iterative: start with simple requests, review the model's responses, and refine the prompt until the output meets the organisation's needs. This human-guided process ensures that LLMs are used effectively, reducing the risk of errors and helping security teams make better use of available data [30].

3. Related Work

LLMs, as a tool for threat hunting and intelligence, are still in its infancy. A study by Liu et al. [31] evaluated the first public version of ChatGPT against traditional machine learning techniques for Vulnerability Description Mapping (VDM). The researchers used public Common Vulnerability Exposures (CVE) datasets as the basis for a testing framework. The LLM received both strong and weak type prompts for the same queries, and the first task was to match a vulnerability description to a valid Common Weakness Enumeration ID (CWE-ID) in the CVE datasets. The second was to match the prompted strong and weak vulnerability descriptions to an ATT&CK technique ID. The study highlighted that while the LLM performed poorly on the second ATT&CK ID, the second task results showed promise, with the LLM achieving accuracies over 80% by the 3rd iteration of the prompt. Nevertheless, the study is focused only on the first version of ChatGPT, and the comparison with traditional AI techniques was only possible for the ATT&CK IDs.

The work of Moskal et al. [32] also explored the application of ChatGPT as an automated VDM tool, with an implementation that relies on prompt engineering to interpret command line outputs and generate executable commands. They set up the automated agent to conduct a red-team campaign in a Dockerised sandbox environment and evaluated the LLM's ability to perform VDM tasks on a set of ten vulnerable services. The research highlights the potential for LLMs as an autonomous tool for detecting vulnerabilities. However, their design is around ChatGPT model version 3.5 with a training cutoff date of 2021. This, combined with the constrained sandbox environment, provides a limited scope in the application area of the study.

The research by Schwartz et al. [33] proposed a new framework, LLMCloudHunter, based on LLM prompt-engineering for automating the creation of Sigma detection rule candidates from the Open-source cyber threat intelligence (OSCTI) database. Their framework is based on the ChatGPT-4o model and goes through three phases of preprocessing, paragraph-level processing and OCSTI-level processing to refine the ruleset. Their work demonstrates the advantages of modern versions of ChatGPT in threat hunting by combining them with traditional methods of processing unstructured data to increase the quality of the prompts with manual interventions. The evaluation results show promise as a tool for automated Sigma rule generation that can be directly integrated into security information and event management (SIEM) systems. Their framework is mostly limited because of its confinement to using only paragraph-structured OSCTIs.

4. Methodology

Threat hunting typically relies on experienced security professionals who collect threat intelligence, develop hypotheses, and analyse log data to detect activities that evade automated security measures. This study's main question is whether LLMs can produce effective search queries for individuals without specialised security expertise. The methodology assumes that core threat hunting steps, such as reviewing threat intelligence and forming hypotheses, have already been carried out. The focus is on whether LLM-generated queries can detect ten different LotL techniques. These techniques are executed on a host similar to those found in commercial environments, and the resulting logs are ingested into security tools. To generate queries for each of these monitoring tools, two LLMs were given three progressively detailed prompts per tool. Each prompt was allowed up to three iterations if the initial query was unsuccessful, resulting in a maximum of nine queries per monitoring tool for each technique. Across all ten techniques, three security tools, and two LLMs, a total of 180 queries were produced and evaluated. A query was deemed successful if it correctly identified the execution of the LotL technique within the logs, regardless of the outcome of the executed command itself. The testing framework is divided into four phases, Preparation, Execution, Analysis, and Discussion, which is illustrated in the workflow Figure 2. The steps involved are as follows:

1. Execute the ten LotL techniques.
2. Collect and ingest the resulting data into security tools.
3. Verify that the event has been logged and is accessible for analysis.
4. Generate three queries for each security tool, using two different LLMs and prompts of increasing detail.
5. Document and analyse the results to assess the effectiveness of LLM-generated queries.

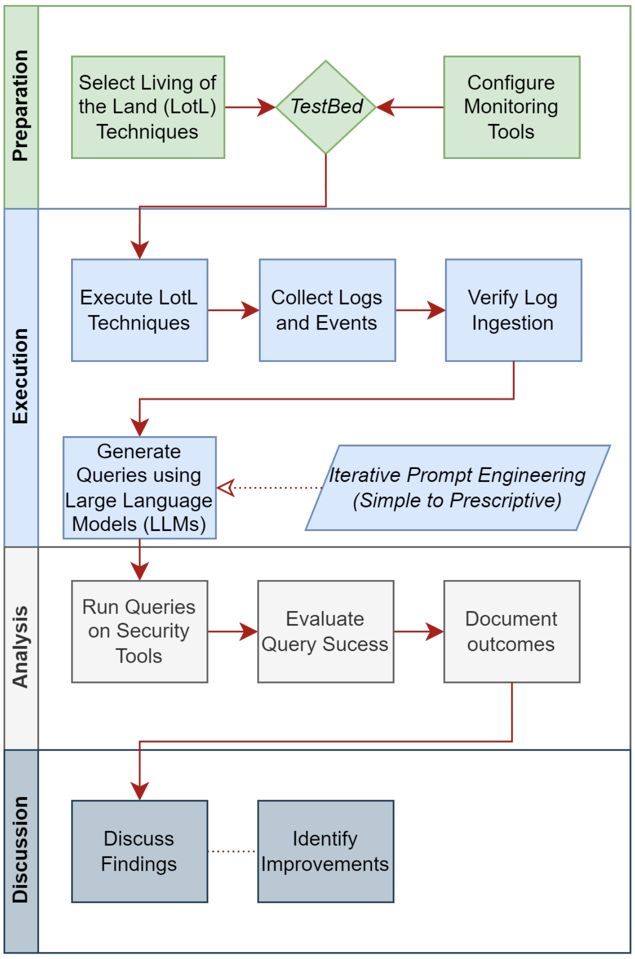

Preparation
Select Living of the Land (LotL) Techniques
TestBed
Configure Monitoring Tools
Execution
Execute LotL Techniques
Collect Logs and Events
Verify Log Ingestion
Generate Queries using Large Language Models (LLMs)
Iterative Prompt Engineering (Simple to Prescriptive)
Analysis
Run Queries on Security Tools
Evaluate Query Sucess
Document outcomes
Discussion
Discuss Findings
Identify Improvements

Figure 2. Threat hunting experiment workflow.

4.1. Limitations

As noted in Section 2.5, prompt creation and refinement can be influenced by the biases and skill level of the person designing the prompts. Prompt engineering also tends to be iterative and can require many adjustments. To maintain consistency across all techniques, LLMs, and tools tested, this study limits the number of refinements to three for each query.

4.2. Preparation Phase

During the preparation phase, the testing environment is configured to resemble real-world conditions as closely as possible. Tasks include choosing the LotL techniques, selecting the appropriate security tools, configuring log collection, and setting up a virtual host for running tests.

4.2.1. Technique Selection

- LOLBAS Project [34]: The Living Off The Land Binaries and Scripts (LOLBAS) repository catalogues legitimate system utilities that attackers commonly misuse. It is regularly updated to account for new discoveries.
- Atomic Red [35]: Red Canary's Atomic Red Team library provides ready-made adversarial tests mapped to the MITRE ATT&CK framework. These tests require minimal setup and help evaluate whether LLM-generated queries can detect malicious activities.
- MITRE ATT&CK [36]: This framework categorises attacker tactics, techniques, and procedures (TTPs) for each stage of an attack. It assists in attributing LotL techniques

to real-world threat actors, ensuring that the study focuses on practical and timely use cases.

4.2.2. Testbed Environment

For this study, three virtual machines running Windows 11 Enterprise were created using VMware Workstation 17 Pro. One of these hosts had a Splunk Universal Forwarder and the Elastic Agent installed, forwarding logs to Splunk hosted on a separate monitoring server and to an Elastic cloud instance. This same host also employed Microsoft's native security tools, and Sysmon was added to capture more detailed event data.

Another Windows 11 host was enrolled in Microsoft Defender for Endpoint and covered by an Office 365 license plus a Microsoft for Endpoint Plan 2 license. These licenses provided the necessary data sources and threat hunting capabilities.

A third machine served as a central monitoring server, running Splunk Enterprise as an on-premises solution. It was responsible for ingesting logs, configuring Splunk, and supporting threat hunting activities.

Two hosts were designated as *victims* , which are displayed in Figure 3 as Hosts 1 and 2. Certain security exceptions were set to permit the execution of some commands, given that the native security tools had flagged them as malicious. The goal was not to evaluate how effective these commands were but rather to test the ability of LLMs to generate queries capable of detecting them.

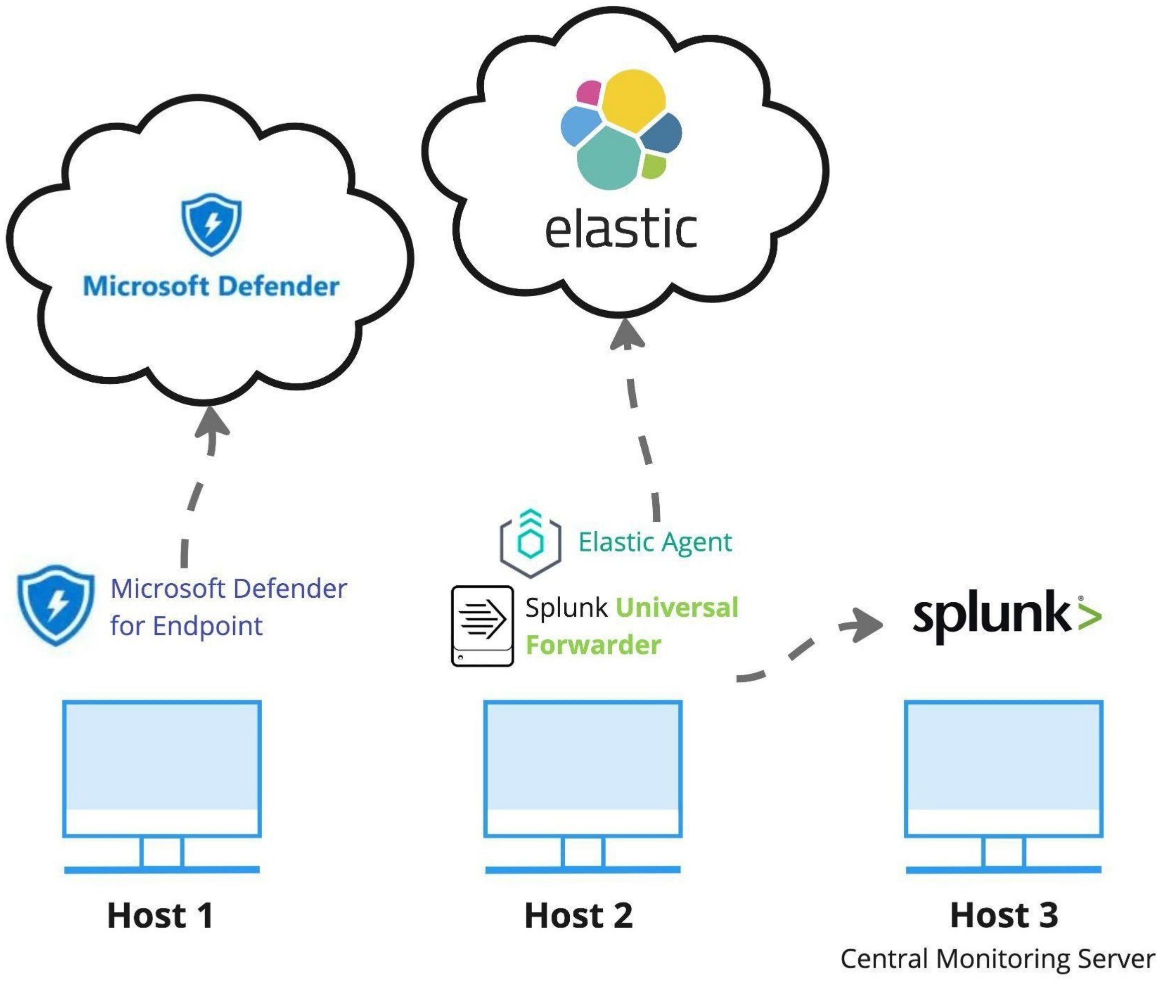

Microsoft Defender
elastic
Elastic Agent
Microsoft Defender for Endpoint
Splunk Universal Forwarder
splunk>
Host 1
Host 2
Host 3
Central Monitoring Server

Figure 3. Testbed environment setup.

4.2.3. AI Tools

Gemini 1.0 [37] and ChatGPT 3.5 [22] were selected for this study because of their generative capabilities and demonstrated performance in natural language processing tasks. Both models were publicly accessible when the experiment occurred and operated independently of any additional tooling to support reproducibility.

4.2.4. LotL Technique Selection

Ten techniques from LOLBAS [34] were chosen for testing. These were reviewed against the MITRE ATT&K framework [36] to determine which threat actors, if any, leverage those LotL techniques. The number and type of techniques were arbitrary. Table 1 shows a concise list of the techniques selected for testing.

Table 1. List of LotL techniques used for testing.

Binary	MITRE ATT&CK Technique ID	Threat Actors
SyncAppvPublishingServer	T1216 – System Script Proxy Execution	Unconfirmed
comsvcs.dll	T1003.001 – OS Credential Dumping: LSASS Memory	Magic Hound, Sandworm
WMI	T1490 – Inhibit System Recovery	BlackCat, Meteor, WannaCry, Wizard Spider
net1.exe	T1136.001 – Create Account: Local Account	Kimsuky
bitsadmin.exe	T1197 – BITS Jobs	APT41, Bazar, Egregor, Leviathan, Wizard Spider, Cobalt Strike, Patchwork, ProLock
conhost.exe	T1202 – Indirect Command Execution	Unconfirmed
cipher.exe	T1485 – Data Destruction	MegaCortex (ransomware), RansomEXX
sc.exe	T1489 – Service Stop	Unconfirmed
findstr	T1552.001 – Unsecured Credentials: Credentials In Files	Unconfirmed
Certificates	T1649 – Steal or Forge Authentication Certificates	APT29

4.2.5. Query Selection

The prompts used in this experiment were predefined to ensure consistency across both security tools being evaluated. An essential consideration when developing these prompts was to keep them deliberately non-technical, closely emulating queries that might be created by individuals without prior threat hunting or security expertise. During testing, it was observed that partial query matches from LLM-generated prompts, although potentially useful in theory, rarely occurred. While a partial match could have indicated some level of success, it might also have led to multiple false positives, thus requiring technical expertise to differentiate between true and false positives. This highlights the importance of query accuracy, highlighting the necessity for LLM-generated queries to be precise enough to minimise false positives, especially when used by non-technical personnel.

4.3. Execution Phase

During this phase, each of the ten LotL techniques is executed on the designated victim machines. To confirm that each command is being logged and ingested by the chosen monitoring tools, a control query is first run on each tool. Once the presence of log entries is verified, three prompts are provided to each AI tool, as shown in Figure 4, in order to generate a query that can detect the LotL activity in the logs. Each prompt is increasingly prescriptive, and they are given within the same session so that the AI tools can build on any context or previous attempts.

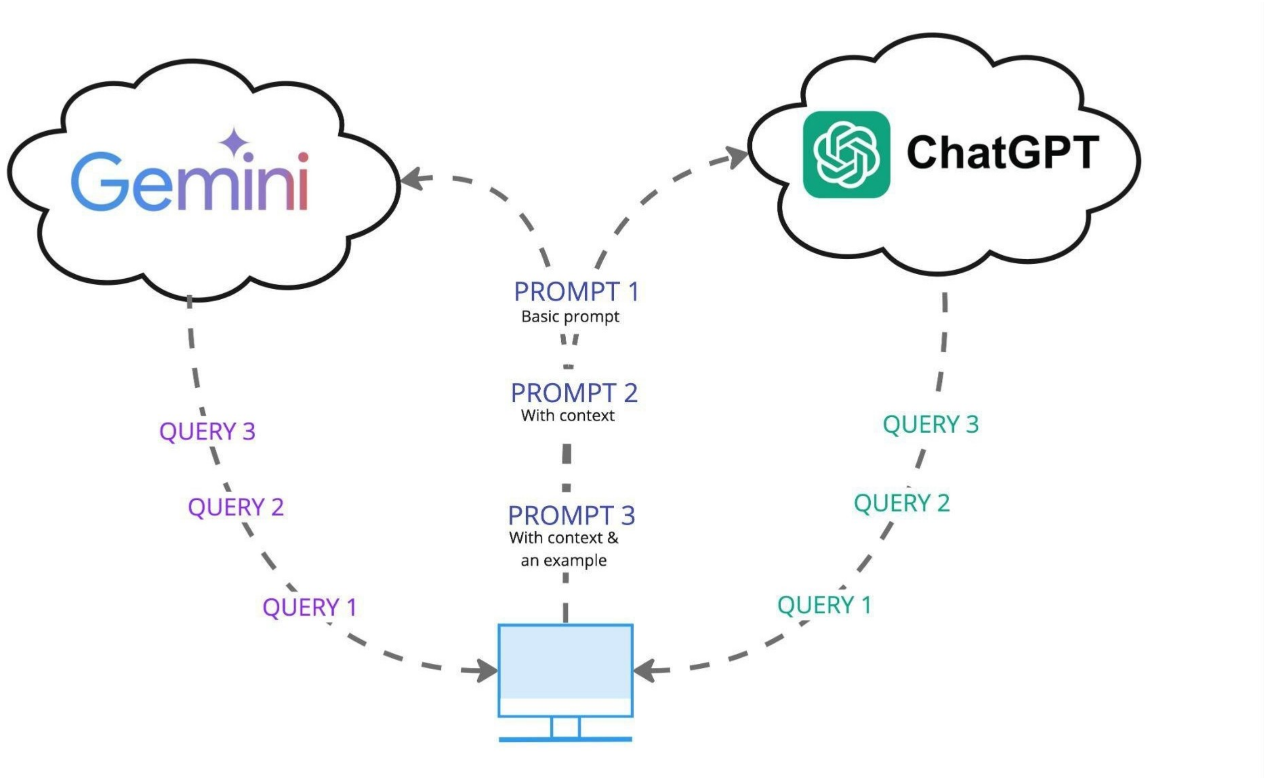


Figure 4. Test prompts and queries.

After the queries are generated, they are tested using the respective security tools. The results are then examined to determine whether a log entry corresponding to the executed LotL technique is returned.

4.4. Analysis Phase

Following the execution phase, the results are evaluated to determine whether each generated query successfully returned the relevant log entry. In the event of a syntax error, a false positive, or a query that yields no results, the outcome is flagged as a failure. Situations where the result is unclear or the data do not align with the success or failure criteria are marked as undetermined.

Three categories of outcomes are defined:

- **Success:** The query accurately retrieves the log entry for the executed command.
- **Failure:** The query either produces an error, yields no relevant data, or shows a false positive.
- **Undetermined:** The outcome cannot be clearly classified as either a success or a failure.

5. Results

Figures 5–7 present a summary of the findings for Microsoft Defender, Splunk, and Elastic/Kibana, respectively. These summaries reflect whether each query generated by the AI tools was successful in retrieving the expected log entries.

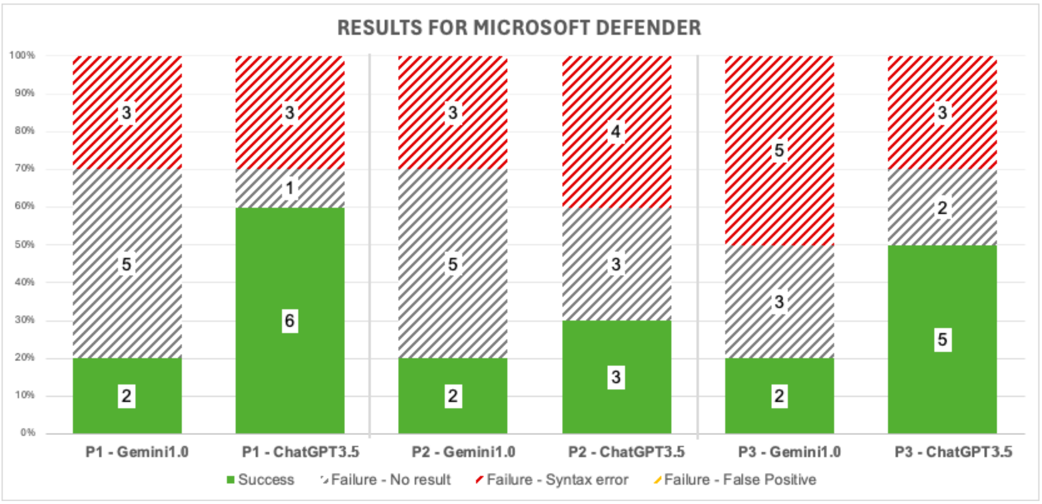


Figure 5. Results for Microsoft Defender.

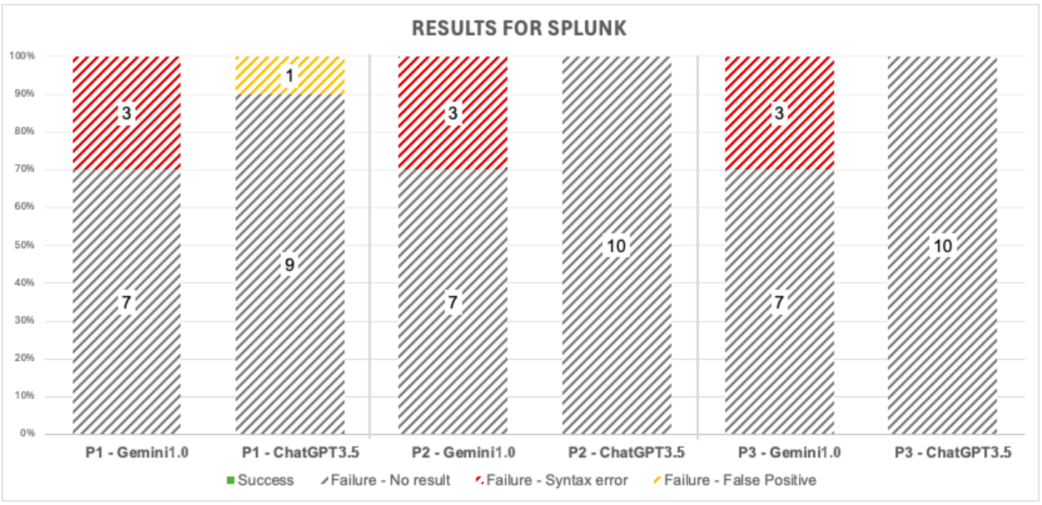


Figure 6. Results for Splunk.

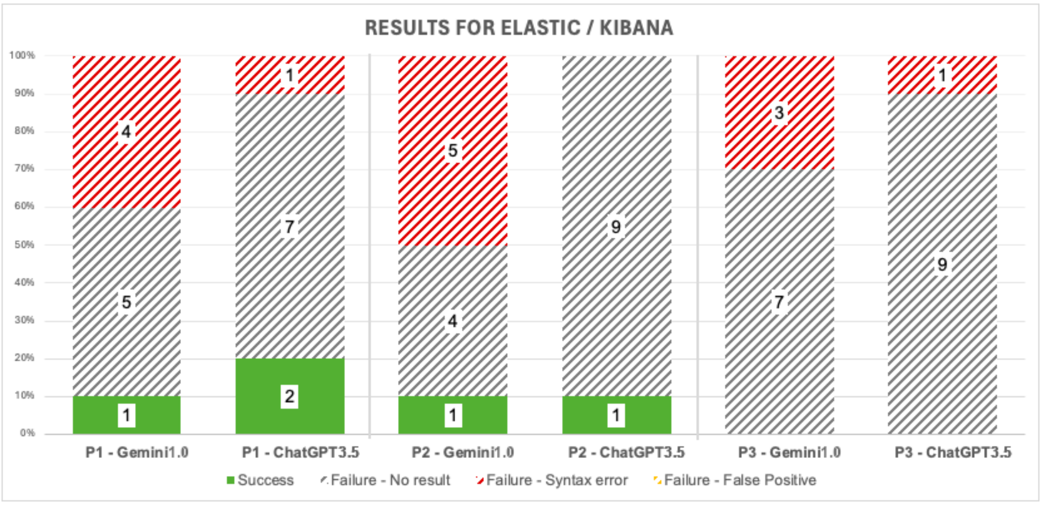


Figure 7. Results for Elastic/Kibana.

Analysis of Results

In contrast to initial expectations, Microsoft Defender queries using Kusto Query Language (KQL) achieved the highest success rate, while Splunk queries returned no successful results. Moreover, ChatGPT queries, particularly for Microsoft Defender, tended to be more accurate than those from other AI tools.

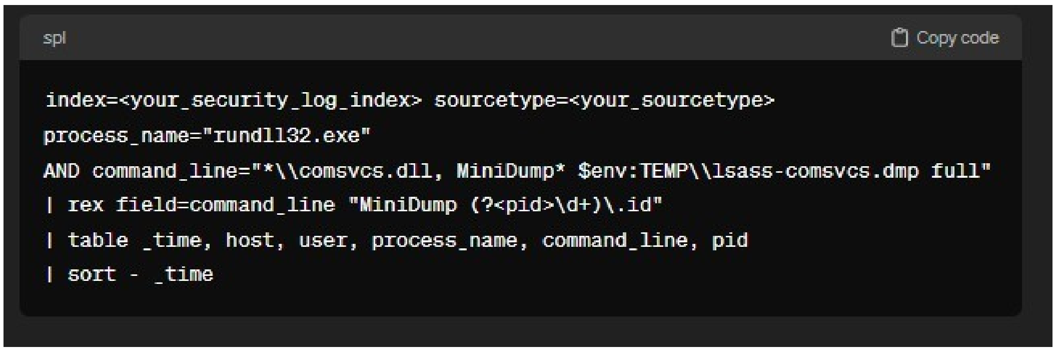
An additional, unanticipated finding was that more detailed or context-rich prompts did not always yield better results. In some cases, simpler prompts (Prompt 1) matched or exceeded the performance of more complex prompts, especially for Microsoft Defender and Elastic/Kibana.

Most failures resulted from queries targeting incorrect indexes, tables, or data sources, leading to no returned results. This suggests that knowledge of each platform's data structures and indexing is essential for creating effective queries, regardless of how advanced or detailed the prompt may be.

6. Discussion

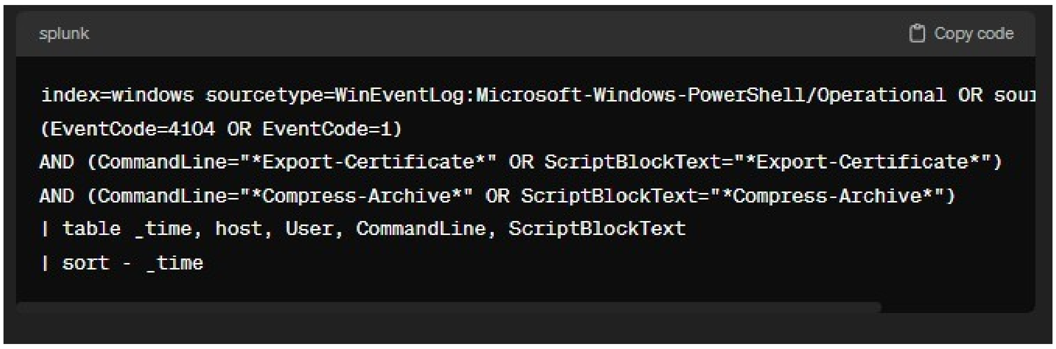
As outlined previously, this study involved generating three queries for each of the three monitoring tools (Microsoft Defender, Splunk, and Elastic/Kibana), using two distinct LLMs (Gemini and ChatGPT) for each of the LotL techniques described. This produced a total of 180 queries. The prompts were written under the assumption that the individual requesting the queries had a non-technical background, with basic familiarity with threat hunting and the tools in question. Additionally, the prompts were kept relatively generic to reduce any potential bias when guiding the LLMs.

In many instances, the LLMs not only provided queries but also offered supplementary advice about running threat hunts and using security tools. Furthermore, some queries returned by the LLMs requested user input before execution, as shown in Figures 8 and 9. These examples highlight the variability in how different prompts and contexts can influence LLM output.



```
index=<your_security_log_index> sourcetype=<your_sourcetype>
process_name="rundll32.exe"
AND command_line="*\\comsvcs.dll, MiniDump* $env:TEMP\\lsass-comsvcs.dmp full"
| rex field=command_line "MiniDump (?<pid>\d+)\.id"
| table _time, host, user, process_name, command_line, pid
| sort - _time
```

Figure 8. Snippet from ChatGPT Splunk query, an example of a query requesting the user to input index and sourcetype.

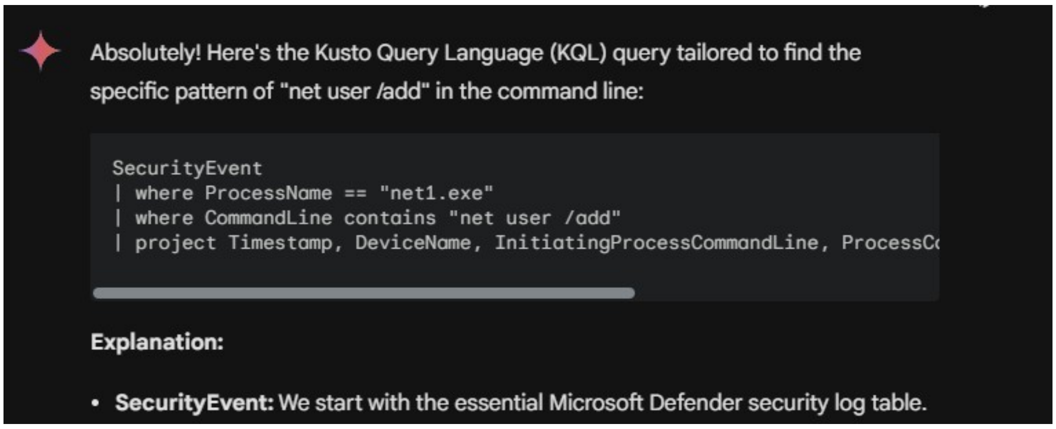


```
index=windows sourcetype=WinEventLog:Microsoft-Windows-PowerShell/Operational OR sou
(EventCode=4104 OR EventCode=1)
AND (CommandLine="*Export-Certificate*" OR ScriptBlockText="*Export-Certificate*")
AND (CommandLine="*Compress-Archive*" OR ScriptBlockText="*Compress-Archive*")
| table _time, host, User, CommandLine, ScriptBlockText
| sort - _time
```

Figure 9. Snippet from ChatGPT Splunk query, an example of a query with pre-populated index and sourcetype.

Overall, the queries generated by the LLMs were largely unsuccessful in retrieving valid and relevant results during the experiment. Most queries either failed to produce meaningful data or contained syntax errors that prevented their successful execution. Although some level of failure had been anticipated due to known limitations in the reasoning abilities of LLMs, the actual failure rate significantly exceeded initial expectations. The difficulty in understanding contextual nuances and the varying structures of security tools' data indexing methods appeared to significantly contribute to these unsuccessful outcomes.

A key factor in the inconsistency and frequent query failures relates to the distinct ways each security tool stores, indexes, and retrieves data. For instance, Splunk's query failures were primarily due to mismatches in indexing structures, causing discrepancies between the LLM-generated queries and the actual data environment. Conversely, Microsoft Defender may have been slightly more compatible with LLM-generated queries due to better alignment between the LLM's training data and the tool's structure or syntax conventions. Additionally, inconsistencies were evident even within single LLM sessions. Some queries included specific, pre-filled fields that did not align with actual organisational indices or data sources, while others required manual input, placing additional burdens on users, which would defeat the objective of LLMs used as an easy-to-access tool for non-experts. Furthermore, incorrect tool-specific commands or arguments in the queries, illustrated by the Gemini-generated query in Figure 10, led directly to execution errors and query failures as displayed in the response of Microsoft Defender in Figure 11.

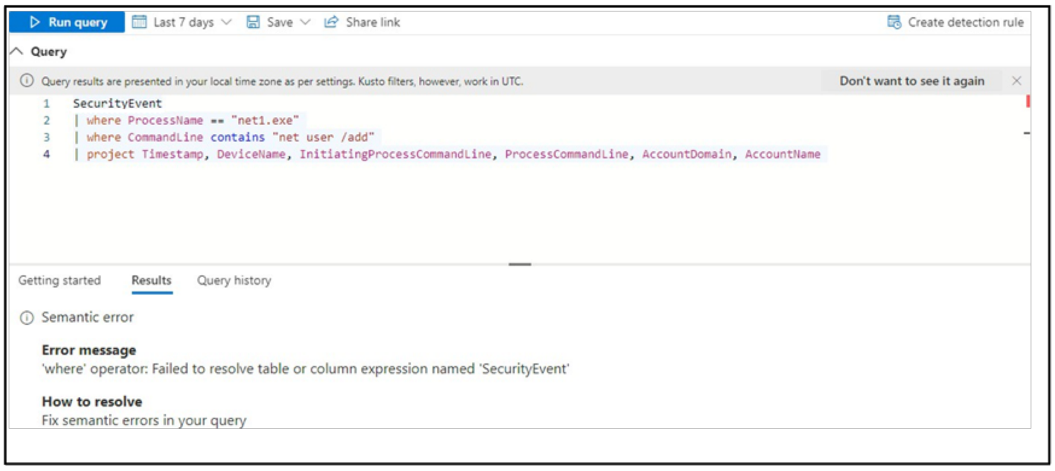

Absolutely! Here's the Kusto Query Language (KQL) query tailored to find the specific pattern of "net user /add" in the command line:

```
SecurityEvent
| where ProcessName == "net1.exe"
| where CommandLine contains "net user /add"
| project Timestamp, DeviceName, InitiatingProcessCommandLine, ProcessC
```

Explanation:

- **SecurityEvent:** We start with the essential Microsoft Defender security log table.

Figure 10. Gemini generates a query to hunt for the creation of a new user, the query attempts to pull data from the SecurityEvent table.



Query results are presented in your local time zone as per settings. Kusto filters, however, work in UTC.

```
SecurityEvent
| where ProcessName == "net1.exe"
| where CommandLine contains "net user /add"
| project Timestamp, DeviceName, InitiatingProcessCommandLine, ProcessCommandLine, AccountDomain, AccountName
```

Semantic error

Error message
'where' operator: Failed to resolve table or column expression named 'SecurityEvent'

How to resolve
Fix semantic errors in your query

Figure 11. Microsoft Defender Advanced Threat Hunting error indicating SecurityEvent is not present.

A comparison could be drawn between the two models, Gemini and ChatGPT, while dissecting the results. Specifically, ChatGPT produced queries with higher accuracy in syntax and structure than Gemini, likely attributed to differences in their training data and underlying architecture. Gemini's outputs had a higher reliance on user input and examples, leading to overly specific queries that could not address the attacks on a general level. In contrast, ChatGPT often produced highly generalised queries that were too vague and, like Gemini, required extensive user input to have any practical utility. The model's differential performance suggests varying levels of contextual understanding and responsiveness to prompt complexity, with ChatGPT faring slightly better results. However, both models appear to have a strong need for user intervention to achieve usability in their queries.

To examine this notion further, the user provided an example of the command they wished to detect in the third set of prompts. While this was not meant to be a rigid instruction, both LLMs constructed queries that exactly matched the sample rather than generalising it. This further indicates that LLMs tend to rely heavily on the specific text given in prompts instead of devising broader detection patterns, such as using regular expressions.

To produce a usable query for threat hunting, a user would need to write a highly detailed prompt or iterate through multiple refinements. This indicates that substantial knowledge is required to guide LLMs towards accurate queries. Even if an LLM can draft a query, the user must still understand the context and structure of the data, along with the logging environment, to ensure that the query uses the correct fields and sources. In practice, if users already possess the required knowledge of the tools, they might generate these queries themselves without relying on an LLM. Consequently, LLMs do not yet offer a clear advantage over human-generated queries in many threat hunting scenarios.

The overall outcomes deviated from the initial hypothesis. While some degree of query failure was foreseen, the rate of failure in Splunk, in particular, was unexpectedly high. Additionally, the nature of these failures is concerning: a user might assume their search was successful if it returned no results, mistakenly concluding there was no malicious activity. The varying success rates across tools were also unexpected, with Microsoft Defender consistently yielding more positive outcomes than Splunk. More intricate prompts did not necessarily correlate with better success, and in some cases, simpler prompts (Prompt 1) performed equally well or better.

A second round of testing was required because, during the initial trials, certain events were not recorded in the log data, skewing the results. Once these logging issues were addressed, the tests were repeated with more consistent data ingestion.

Given that LLMs are increasingly promoted as a means of enhancing work efficiency, these findings suggest that using them for specialised tasks such as threat hunting still poses challenges. Effective use of LLMs would require comprehensive training in both prompt engineering and the specific query language of the security tool in question. Without this knowledge, iterating prompts to reach a functional query may be more time-consuming than manually composing a query. Consequently, LLMs should not be viewed as a replacement for thorough threat hunting procedures or for the technical skills needed to interpret and respond to potential threats. Instead, and as LLMs develop, they could be leveraged as tools for assisting users in query writing rather than fully automating the process.

7. Conclusions

Threat hunting is a proactive measure that helps organisations defend against complex threats. It calls for specialist skills to recognise indicators of compromise, gather relevant data, and decide where to look for malicious activity. The shortage of qualified security personnel makes implementing and maintaining threat hunting programmes difficult, especially for smaller organisations with limited budgets. This challenge is compounded by threat actors' reliance on LotL techniques, which mask malicious activity among legitimate system processes. Moreover, as organisations transition to Industry 4.0 and integrate IoT devices into their networks, the expanded attack surface further increases the importance of skilled threat hunters who can detect subtle signs of intrusion.

This study examined whether LLMs could generate effective queries for threat hunting in a way that would help less experienced or non-security professionals. The purpose was for the LLMs to work with a certain level of autonomy to demonstrate their efficacy as advisors to non-security teams. Ten LotL techniques were tested in a lab setting, and two LLMs were given prompts to create queries for commonly used security tools. Although the research met its objectives, the findings showed a high rate of failed or inconsistent queries due to issues like syntax errors and zero returned results. Thus, LLMs alone are not yet a reliable source of high-quality queries for threat hunting and should not replace structured processes or the essential training required for effective cybersecurity practices.

8. Future Work

Future research should examine how regular LLM use by novice threat hunters affects their ability to learn essential threat hunting skills, comparing their progress with experienced practitioners. Additionally, it is important to investigate whether frequent interaction with LLMs enhances junior analysts' professional skills or if it inadvertently limits their development of analytical thinking and domain expertise. Studies could provide insights into the lasting impacts of LLM usage on analysts' skill retention and confidence.

Effective LLM use also demands specialised training in prompt engineering and security-specific query languages. Further research should explore fine-tuning LLMs for cybersecurity prompt comprehension to improve their accuracy and reliability. Investigations into hybrid human-in-the-loop methods, combining human oversight with LLM outputs, could establish whether this approach offers improved accuracy compared with fully automated methods. In addition, a direct comparison of newer models of LLMs against human-generated queries from industry experts would help to scrutinise the supposed higher capabilities of this generation of models.

Moreover, comparing LLM-based approaches against traditional AI-driven security automation would clarify the advantages and limitations of each. Such comparisons would identify contexts in which LLMs perform better or where conventional automation methods remain more suitable.

Finally, research must address existing limitations of LLMs, including biases, hallucinations, and interpretability issues. Targeted efforts to mitigate these problems through improved prompt engineering, fine-tuning, and enhanced transparency could significantly improve the practical use of LLMs in cybersecurity.

Author Contributions: Conceptualisation, A.K., D.K., and N.P.; methodology, A.K.; software, A.K.; validation, A.K., D.K., and N.P.; data curation, A.K.; writing—original draft preparation, A.K. and D.K.; writing—review and editing, D.K., W.J.B., S.U.J., J.A., I.P., and N.P.; supervision, W.J.B., I.P., and N.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Dataset available upon reasonable request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yao, Y.; Duan, J.; Xu, K.; Cai, Y.; Sun, Z.; Zhang, Y. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confid. Comput.* **2024**, *4*, 100211. [CrossRef]
2. Rawindaran, N.; Jayal, A.; Prakash, E. Machine learning cybersecurity adoption in small and medium enterprises in developed countries. *Computers* **2021**, *10*, 150. [CrossRef]

3. Barr-Smith, F.; Ugarte-Pedrero, X.; Graziano, M.; Spolaor, R.; Martinovic, I. Survivalism: Systematic analysis of windows malware living-off-the-land. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA , 24–27 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1557–1574.
4. Pitropakis, N.; Pikrakis, A.; Lambrinoudakis, C. Behaviour reflects personality: detecting co-residence attacks on Xen-based cloud environments. *Int. J. Inf. Secur.* **2015**, *14*, 299–305. [CrossRef]
5. Ussath, M.; Jaeger, D.; Cheng, F.; Meinel, C. Advanced persistent threats: Behind the scenes. In Proceedings of the 2016 Annual Conference on Information Science and Systems (CISS), Princeton, NJ, USA, 16–18 March 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 181–186.
6. Aldauiji, F.; Batarfi, O.; Bayousef, M. Utilizing cyber threat hunting techniques to find ransomware attacks: A survey of the state of the art. *IEEE Access* **2022**, *10*, 61695–61706.
7. Chidukwani, A.; Zander, S.; Koutsakis, P. A survey on the cyber security of small-to-medium businesses: challenges, research focus and recommendations. *IEEE Access* **2022**, *10*, 85701–85719. [CrossRef]
8. Rizvi, S.; Orr, R.; Cox, A.; Ashokkumar, P.; Rizvi, M.R. Identifying the attack surface for IoT network. *Internet Things* **2020**, *9*, 100162. [CrossRef]
9. CrowdStrike. WIZARD SPIDER Update: Resilient, Reactive and Resolute. 2020. Available online: https://www.crowdstrike.com/en-us/blog/wizard-spider-adversary-update/ (accessed on 22 December 2024).
10. IBM. What Is Threat Hunting? 2025. Available online: https://www.ibm.com/think/topics/threat-hunting (accessed on 8 January 2024).
11. Gunter, D.; Seitz, M. A Practical Model for Conducting Cyber Threat Hunting. *Sans. org* **2018**. Available online: https://www.sans.org/white-papers/38710/ (accessed on 8 January 2024).
12. Fuchs, M.; Lemon, J. In *Sans 2019 Threat Hunting Survey: The Differing Needs of New and Experienced Hunters*; SANS Institute Information Reading Room: Rockville Pike, MD 20852, USA, 2019.
13. Blažič, B.J. The cybersecurity labour shortage in Europe: Moving to a new concept for education and training. *Technol. Soc.* **2021**, *67*, 101769. [CrossRef]
14. Central Digital and Data Office. Detecting the Unknown: A Guide to Threat Hunting. 2019. Available online: https://hodigital.blog.gov.uk/wp-content/uploads/sites/161/2020/03/Detecting-the-Unknown-A-Guide-to-Threat-Hunting-v2.0.pdf (accessed on 19 December 2024).
15. Brown, D. Preventing Living Off the Land Attacks. 2021. Available online: https://www.sans.org/white-papers/39450/ (accessed on 20 December 2024).
16. Villalón-Huerta, A.; Ripoll-Ripoll, I.; Marco-Gisbert, H. Key requirements for the detection and sharing of behavioral indicators of compromise. *Electronics* **2022**, *11*, 416. [CrossRef]
17. Vaswani, A. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017;
18. Brown, T.B. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.
19. McCoy, R. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. *arXiv* **2019**, arXiv:1902.01007.
20. Bender, E.M.; Gebru, T.; McMillan-Major, A.; Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event, Canada, 3–10 March 2021; pp. 610–623.
21. Bender, E.M.; Koller, A. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics; Jurafsky, D., Chai, J., Schluter, N., Tetreault, J., Eds.; Online, 5–10 July 2020; pp. 5185–5198. [CrossRef]
22. Floridi, L.; Chiriatti, M. GPT-3: Its nature, scope, limits, and consequences. *Minds Mach.* **2020**, *30*, 681–694.
23. Mittelstadt, B.D.; Allo, P.; Taddeo, M.; Wachter, S.; Floridi, L. The ethics of algorithms: Mapping the debate. *Big Data Soc.* **2016**, *3*, 2053951716679679. [CrossRef]
24. Microsoft. Introducing Microsoft 365 Copilot—Your Copilot for Work. 2023. Available online: https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/ (accessed on 23 December 2024).
25. Crowdstrike. Introducing Charlotte AI, CrowdStrike's Generative AI Security Analyst: Ushering in the Future of AI-Powered Cybersecurity. 2023. Available online: https://www.crowdstrike.com/blog/crowdstrike-introduces-charlotte-ai-to-deliver-generative-ai-powered-cybersecurity/ (accessed on 23 December 2024).
26. Google. Prompt Engineering for Generative AI. 2023. Available online: https://developers.google.com/machine-learning/resources/prompt-eng (accessed on 23 December 2024).
27. Reynolds, L.; McDonell, K. Prompt programming for large language models: Beyond the few-shot paradigm. In Proceedings of the Extended abstracts of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama Japan, 8–13 May 2021; pp. 1–7.

28. Chen, B.; Zhang, Z.; Langrené, N.; Zhu, S. Unleashing the potential of prompt engineering in Large Language Models: A comprehensive review. *arXiv* **2023**, arXiv:2310.14735.
29. Subramanian, K. Introducing the Splunk platform. In *Practical Splunk Search Processing Language: A Guide for Mastering SPL Commands for Maximum Efficiency and Outcome*; Apress: Berkeley, CA, USA, 2020; pp. 1–38.
30. Sindiramutty, S.R. Autonomous Threat Hunting: A Future Paradigm for AI-Driven Threat Intelligence. *arXiv* **2023**, arXiv:2401.00286.
31. Liu, X.; Tan, Y.; Xiao, Z.; Zhuge, J.; Zhou, R. Not the end of story: An evaluation of ChatGPT-driven vulnerability description mappings. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2023, Toronto, ON, Canada, 9–14 July 2023; pp. 3724–3731.
32. Moskal, S.; Laney, S.; Hemberg, E.; O'Reilly, U.M. Llms killed the script kiddie: How agents supported by large language models change the landscape of network threat testing. *arXiv* **2023**, arXiv:2310.06936.
33. Schwartz, Y.; Benshimol, L.; Mimran, D.; Elovici, Y.; Shabtai, A. Llmcloudhunter: Harnessing llms for automated extraction of detection rules from cloud-based cti. *arXiv* **2024**, arXiv:2407.05194.
34. LOLBAS-Project. LOLBAS-Project on GitHub. Available online: https://github.com/LOLBAS-Project/LOLBAS (accessed on 20 December 2024).
35. Red Canary. Atomic Red Team. Available online: https://github.com/redcanaryco/atomic-red-team (accessed on 20 December 2024).
36. MITRE. MITRE ATT&CK Framework. Available online: https://attack.mitre.org/ (accessed on 20 December 2024).
37. Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A.M.; Hauth, A.; Millican, K.; et al. Gemini: A family of highly capable multimodal models. *arXiv* **2023**, arXiv:2312.11805.