Evaluating LLMs for the Automated Generation of Operational Detection Rules in Enterprise EDR Environments

Ioannis Konstantaras , Efstratios Chatzoglou , Konstantinos E. Kampourakis * , Georgios Kambourakis

Posted Date: 26 March 2026

doi: 10.20944/preprints202603.1994.v1

Keywords: detection engineering; endpoint detection and response; LLM; threat intelligence operationalization

 doi:10.20944/preprints202603.1994.v1

Article

Evaluating LLMs for the Automated Generation of Operational Detection Rules in Enterprise EDR Environments

Ioannis Konstantaras [1], Efstratios Chatzoglou [1], Konstantinos E. Kampourakis [2,*] and Georgios Kambourakis [1]

[1] Department of Information & Communication Systems Engineering, University of the Aegean, 83200 Karlovasi, Greece
[2] Norwegian University of Science and Technology, 2802 Gjøvik, Norway
* Correspondence: konstantinos.kampourakis@ntnu.no

Abstract

Modern Endpoint Detection and Response (EDR) platforms, such as Microsoft Defender for Endpoint (MDE), provide sophisticated telemetry but often leave Security Operations Centers (SOC) struggling with a significant detection lag, namely the time required to manually translate emerging threat intelligence into operational logic. This paper presents a systematic empirical study of an LLM-integrated pipeline designed to automate the transformation of structured threat intelligence from OpenCTI into functional Kusto Query Language (KQL) detection rules. By utilizing Large Language Models (LLMs) as a contextual translation layer, we evaluate a framework that maps graph-based STIX metadata directly to proprietary EDR schemas. Our experiments, conducted within a high-fidelity Windows Server 2025 environment, reveal that LLM-augmented rules successfully addressed critical visibility gaps in reconnaissance and early-stage lateral movement where native MDE heuristics remained silent. Importantly, the implementation reduced the intelligence-to-logic latency from an average of 45 minutes of manual engineering to sub-5-minute automated cycles. While the findings identify persistent challenges regarding schema hallucinations, the study concludes that LLM-assisted detection engineering serves as a significant operational force multiplier, enabling defensive postures to evolve at the velocity of the modern threat landscape.

Keywords: detection engineering; endpoint detection and response; LLM; threat intelligence operationalization

1. Introduction

The increasing sophistication and persistence of modern cyber threats have fundamentally reshaped the landscape of organizational security. In particular, Advanced Persistent Threat (APT) groups conduct highly coordinated, multi-stage intrusion campaigns characterized by a low-and-slow approach that evades traditional signature-based defenses. For the purposes of this work, APTs represent the primary adversarial model; their reliance on documented Tactics, Techniques, and Procedures (TTPs) provides the structured behavioral data necessary to evaluate the efficacy of automated rule generation. As organizations expand their attack surface through digital transformation, the ability to rapidly identify Indicators of Compromise (IoCs) and translate them into adversarial behavior patterns has become a foundational requirement for cyber defense [1].

Endpoint Detection and Response (EDR) platforms play an essential role in surfacing malicious behavior across enterprise environments [2]. This study specifically utilizes Microsoft Defender for Endpoint (MDE) as the target defensive environment. The selection of MDE is justified by its dominant position in the enterprise market and its high-fidelity telemetry surface, which includes process, network, and identity-related events. Furthermore, MDE's proprietary Kusto Query Language (KQL) and evolving telemetry schema present a significant technical challenge for LLM-based code

 doi:10.20944/preprints202603.1994.v1

generation. By targeting a black-box commercial EDR rather than a static open-source alternative, this research provides a robust benchmark for an LLM's ability to handle the technical idiosyncrasies and schema-matching requirements of professional-grade security products.

Nevertheless, the effectiveness of such platforms remains contingent upon the quality of detection rules. Because APT techniques are frequently updated to bypass native EDR heuristics, there is a constant operational need for bespoke detection content. Traditionally, crafting these rules is a labor-intensive process, creating a bottleneck in security operations and leading to a detection lag between threat discovery and active defense. The rapid progress in Large Language Models (LLMs) offers a transformative opportunity to automate these workflows by interpreting complex intelligence and generating structured, platform-specific code [3,4].

The main objective of this work is to design and implement an LLM-powered assistant capable of bridging the gap between raw intelligence and operational security controls. To guide this inquiry, we address the following four Research Questions (RQs):

- RQ1: To what extent can an LLM accurately interpret and contextualize graph-based threat intelligence retrieved from OpenCTI?
- RQ2: How reliably can LLM-driven automation map disparate IoCs to specific MITRE ATT&CK techniques?
- RQ3: Is it feasible for an LLM to generate syntactically valid and operationally precise KQL detection rules for MDE without human semantic intervention?
- RQ4: Can such a system materially reduce analyst workload while improving the forensic visibility of silent EDR blocks?

This paper provides several unique contributions to the field of AI-augmented cybersecurity:

- We conduct a systematic empirical evaluation of LLM-generated detection logic across a representative subset of the MITRE ATT&CK matrix, specifically targeting under-detected reconnaissance and lateral movement techniques.
- Unlike related work focusing on simple IoC extraction (hashes/IPs), our approach emphasizes the operationalization of behavioral TTPs to detect early-stage reconnaissance and lateral movement.
- We provide a rigorous before-and-after analysis of detection coverage, quantifying the detection delta provided by LLM-generated rules compared to native MDE heuristics.
- We introduce a custom normalization layer designed to mitigate LLM hallucinations regarding proprietary EDR schemas, addressing the technical reliability of AI-generated security code.

The rest of this paper is organized as follows. The next section reviews relevant studies, while Section 3 and Section 4 describe the methodology and system architecture, respectively. The evaluation results are presented in Section 5, followed by a discussion of the findings in Section 6. Finally, Section 7 concludes the paper and provides avenues for future work.

2. Related Work

Several studies have examined the detection of advanced intrusion techniques within enterprise environments. Notably, Smiliotopoulos et al. [5] offer a comprehensive taxonomy of lateral movement detection methodologies, classifying current solutions into EDR-based, Machine Learning (ML)–based, and graph-based frameworks. Their analysis underscores the inherent difficulty of identifying adversarial activity post-compromise, as lateral movement and credential misuse frequently obfuscate their presence by mimicking legitimate administrative actions. Similarly, Ho et al. [6] propose Hopper, a system that leverages authentication and network telemetry to identify suspicious lateral movement patterns. This work highlights the persistent challenge of distinguishing between malicious operations and benign system behavior.

A significant mass of research has also investigated EDR technologies as a foundational solution for protecting enterprise endpoints. Arfeen et al. [2] provide an overview of EDR systems and their mechanisms for collecting high-fidelity telemetry, such as process execution events, network activity,

 doi:10.20944/preprints202603.1994.v1

and authentication logs. This telemetry enables analysts to identify advanced threats that evade traditional security tools. Moreover, Hassan et al. [7] presented a tactical cyber defense framework that utilizes endpoint telemetry to enhance incident detection and investigation. These studies emphasize the necessity of centralized telemetry analysis for achieving holistic visibility into endpoint operations.

Despite these advancements, recent literature has identified limitations in current EDR-based detection methods. Kaur et al. [8], for example, examine the evolution of EDR technologies, highlighting that the massive volume of telemetry produced by contemporary systems can overwhelm analysts and complicate triaging methodologies. Additionally, Yusof et al. [9] evaluate the efficacy of EDR platforms and demonstrate that identifying subtle signs of compromise often requires bespoke, manually crafted detection rules and expert analysis. These results indicate that optimizing detection engineering workflows remains a significant research challenge.

Researchers often employ the MITRE ATT&CK framework to systematically model adversarial behavior. Al-Sada et al. [10] examine the application of ATT&CK in cybersecurity research, illustrating how the framework facilitates the correlation of adversarial tactics and techniques with defensive detection strategies. Moreover, Rajesh et al. [11] investigate the role of ATT&CK in threat intelligence, demonstrating that modeling attacks as sequences of TTPs enhances the creation of detection mechanisms that align with actual adversary behavior.

Recently, the application of LLMs in cybersecurity has garnered significant interest. Xu et al. [12] present an extensive examination of LLM applications across diverse security domains, encompassing vulnerability detection, malware analysis, and intrusion detection. Zhang et al. [13] also investigate the utility of LLMs in automating security analysis tasks and augmenting human decision-making. However, these studies predominantly concentrate on high-level analytical tasks rather than the technical operationalization of detection engineering.

Building on these foundations, specialized research has emerged regarding the automated synthesis and validation of detection logic. Recent work by Saura et al. [14] and frameworks such as RulePilot [15] underscore a shift toward using autonomous agents to resolve the technical idiosyncrasies of disparate SIEM/EDR schemas. Furthermore, contemporary studies have begun to address the prevention-visibility paradox, a phenomenon where robust EDR enforcement layers thwart attacks through silent blocks, inadvertently stripping the SOC of the forensic telemetry required for actor attribution and intent analysis [16].

In contrast, this paper presents a systematic empirical study of the intelligence-to-logic lifecycle – defined here as the end-to-end technical process of transforming raw, graph-based threat data into operational detection queries for immediate EDR ingestion. While contemporary research by Saura et al. [14] explores LLMs for static security policy mapping, and industry analyses [16] identify the prevention-visibility paradox, our work bridges the gap between theoretical potential and operational reality. By empirically quantifying the detection delta and the significant latency reduction achieved through automated KQL synthesis, this study establishes a methodical benchmark for AI-augmented detection engineering within professional-grade enterprise environments.

3. Methodology

In this paper, we adopt an applied experimental research methodology aimed at evaluating the efficacy of LLMs in supporting detection engineering tasks by transforming structured threat intelligence into operational detection rules for MDE. Rather than focusing on theoretical modeling, the research emphasizes practical implementation, controlled experimentation, and empirical validation within a simulated enterprise security environment.

The methodological process follows a structured sequence beginning with the evaluation and selection of an appropriate Threat Intelligence Platform (TIP). This is followed by the ingestion of structured adversary intelligence, its transformation into detection logic using an LLM, and the deployment of the generated rules within MDE. The final phase consists of executing representative adversarial techniques and evaluating detection outcomes before and after the introduction of custom

 doi:10.20944/preprints202603.1994.v1

rules. Throughout this workflow, the LLM functions as a supporting analytical component, while the human analyst remains responsible for orchestration, deployment, and validation.

3.1. Threat Intelligence Tool Evaluation and Selection

The first step of the methodology involved evaluating several open-source TIPs to determine their suitability for LLM-assisted rule generation. The tools examined included MISP, OpenCTI, and Yeti, while additional platforms such as Prelude and MalPipe were also considered during the initial survey phase [17–22]. Prelude was excluded from further evaluation due to its transition to a proprietary licensing model with limited trial functionality. MalPipe was also excluded following installation issues and the stagnation of its public repository.

MISP, Yeti, and OpenCTI were all found to be capable of managing IoCs and ingesting structured threat intelligence. Yeti, in particular, enabled a relatively straightforward transformation of Sigma rules into KQL via an LLM. However, OpenCTI was ultimately selected as the primary TIP because it offers a more granular and expressive data model. In addition to the capabilities provided by MISP and Yeti, OpenCTI supports graph-based representations of threats, entity relationships, and attack campaigns, maintains native compatibility with STIX 2.1, and provides a robust GraphQL API that enables precise, programmatic querying. These characteristics make OpenCTI particularly suitable for automated integration with an LLM.

3.2. Selection of MITRE ATT&CK Techniques and Attack Scenarios

The attack techniques evaluated in this study were selected based on prevalence and detectability criteria. Preference was given to MITRE ATT&CK techniques that are frequently observed in real-world intrusions but are typically under-detected or inconsistently flagged by out-of-the-box EDR solutions [10]. This approach was deliberately chosen to measure the delta that LLM-generated rules provide beyond native MDE capabilities.

With reference to Table 1, the attack techniques evaluated in this study were selected based on prevalence, detectability, and their direct alignment with the operational profile of APT29. Preference was given to techniques frequently observed in APT29's recent campaigns, which often utilize credential-heavy tactics such as Phishing (T1566) for initial access and OS Credential Dumping (T1003) for post-compromise escalation. By focusing on these specific TTPs, including Kerberoasting (T1558.003) and SMB-based lateral movement (T1021.002), the study targets the prevention-visibility paradox. This ensures the evaluation measures the LLM's ability to restore forensic visibility for behaviors that may be silently blocked or obscured by native EDR heuristics but lack the granular, custom KQL telemetry required for proactive attribution.

As seen in Table 1, the selected techniques span six critical attack categories, ranging from initial access via phishing to lateral movement and credential access within active directory environments. These domains were prioritized because they represent the high-volume, noisy telemetry stages of a sophisticated APT29 operation, where the transition from structured intelligence to operational logic is traditionally labor-intensive for human analysts. Each technique was mapped to a concrete attack scenario implemented using standard offensive tools (e.g., HiddenEye, Impacket, and Atomic Red Team) and executed against our testbed detailed in Section 4. This treatment ensures that the evaluation remains technically grounded, reproducible, and serves as a methodical testbed for measuring the LLM's ability to synthesize logic from the complex behavioral descriptions characteristic of modern state-sponsored threats.

 doi:10.20944/preprints202603.1994.v1

Table 1. Tested attack techniques.

Attack Category	MITRE Technique	Tool Used	Target System
Credential Access (Phishing)	Phishing (T1566)	HiddenEye	Windows 11 Enterprise (user workstation)
Discovery (Account Discovery)	Account Discovery (T1087)	Native tools / scripted enumeration	Windows Server 2025 / Active Directory
Credential Access (Kerberos)	Kerberoasting (T1558.003)	Atomic Red Team / Kerberoast tooling	Windows Server 2025 / Active Directory
Discovery / Lateral Movement (SMB)	Network Share Discovery (T1135) / Remote Services (T1021.002)	SMB enumeration / reverse shell delivery	Windows Server 2025, Windows 11 Enterprise
Credential Access (Dumping)	OS Credential Dumping (T1003)	impacket-secretsdump.py, regsecrets.py, netexec (LSASSY)	Windows Server 2025 (Domain Controller)
Credential Access / Lateral Movement	Pass the Hash (T1550.002)	impacket / netexec	Windows Server 2025, Windows 11 Enterprise

3.3. LLM-Assisted Detection Rule Generation Process & Prompt Engineering

The process of generating detection rules begins by supplying a MITRE ATT&CK technique identifier or name to a custom Python-based assistant. This input triggers a query to the OpenCTI GraphQL API to retrieve the corresponding STIX technique object, which includes its external identifier, associated tactic, supported platforms, and a technical summary.

This structured context is then embedded into a prompt provided to the LLM [23,24]. The prompt is carefully constrained to ensure the model focuses exclusively on generating detection logic rather than explanatory text. The output consists solely of KQL queries intended for execution within MDE. The prompt is grounded in authoritative ATT&CK-aligned intelligence to mitigate hallucinations and ensure alignment with adversarial behavior patterns.

The interaction follows a standardized prompt structure consistent across all evaluated techniques. A system-level instruction defines the role of the model as an experienced SOC analyst and enforces strict output constraints, including the use of valid MDE schema attributes, noise reduction considerations, and the exclusion of conversational filler.

All experiments were conducted using a fixed LLM configuration with a temperature value of 0.0, ensuring deterministic behavior. Although the implementation supports an interactive refinement process, this capability was deliberately excluded from the experimental evaluation to establish a baseline. Instead, the raw output produced by the LLM was used directly, with only minimal syntactic corrections applied to assess the autonomous effectiveness of the generated logic.

3.4. Post-Processing and Validation of Generated KQL

Due to the evolving nature of MDE telemetry schemas, LLM-generated KQL queries may occasionally reference deprecated or inconsistent field names. To address this, a post-processing step was implemented to normalize commonly misused fields, ensure that mandatory projections required by MDE are present, and preserve the logical intent of the query.

After post-processing, each KQL query is executed in the MDE Advanced Hunting interface to verify syntactic correctness and confirm the availability of relevant telemetry. Queries that execute successfully are then converted into custom detection rules and deployed for experimental evaluation.

3.5. Experimental Evaluation Methodology

The evaluation follows a comparative longitudinal design to measure the impact of LLM-generated rules on MDE's visibility. For each selected technique, the corresponding attack was executed twice: first, in a baseline state with no custom rules, and second, after the deployment of the LLM-generated rule.

 doi:10.20944/preprints202603.1994.v1

An attack was considered successfully detected if it resulted in a high-fidelity alert and the creation of an incident within the Microsoft Defender portal. While blocking behavior was documented, it was not required for a successful classification, as the primary objective is improved detection visibility.

3.6. Reproducibility Considerations

The methodology has been developed with reproducibility as a core objective. All essential components—including the TIP, the Python-based assistant, and the specific prompt structure—are thoroughly documented. Nonetheless, external factors such as upstream updates to OpenCTI feeds or variations in underlying model weights may affect the exact results over time. For reasons of reproducibility, the Python-based prototype is publicly available at a GitHub repository [25].

4. System Architecture

This section presents the end-to-end architectural design, deployment layout, and implementation details of the LLM-assisted detection engineering system. As already mentioned, the architecture integrates an LLM-based Python assistant [26] with a virtualized enterprise environment monitored by MDE and supported by an OpenCTI platform enriched with multi-source connectors. The implementation described here forms the experimental foundation for the subsequent evaluation and validation activities.

4.1. System Architecture & Environment

The architecture consists of four interdependent components: the monitored enterprise environment running on Oracle VirtualBox, the MDE cloud backend, the OpenCTI platform enriched by external connectors, and the LLM assistant. The interaction between these components enables analysts to target specific MITRE ATT&CK techniques, retrieve high-context intelligence from OpenCTI, and produce syntactically valid MDE KQL detection queries.

Although OpenCTI synchronizes threat data from MISP feeds, AlienVault OTX, and MITRE ATT&CK, the Python assistant specifically queries MITRE technique objects. The auxiliary connectors indirectly enrich the MITRE graph with intrusion-set metadata and contextual descriptions. This enrichment is a critical design choice, as it influences the content of the technique objects and augments the prompt provided to the LLM without requiring the manual extraction of raw IoCs.

Figure 1 illustrates the overall architecture of the experimental environment, highlighting the interaction between the virtualized enterprise network, the OpenCTI threat-intelligence platform, the LLM-based assistant, and the MDE.

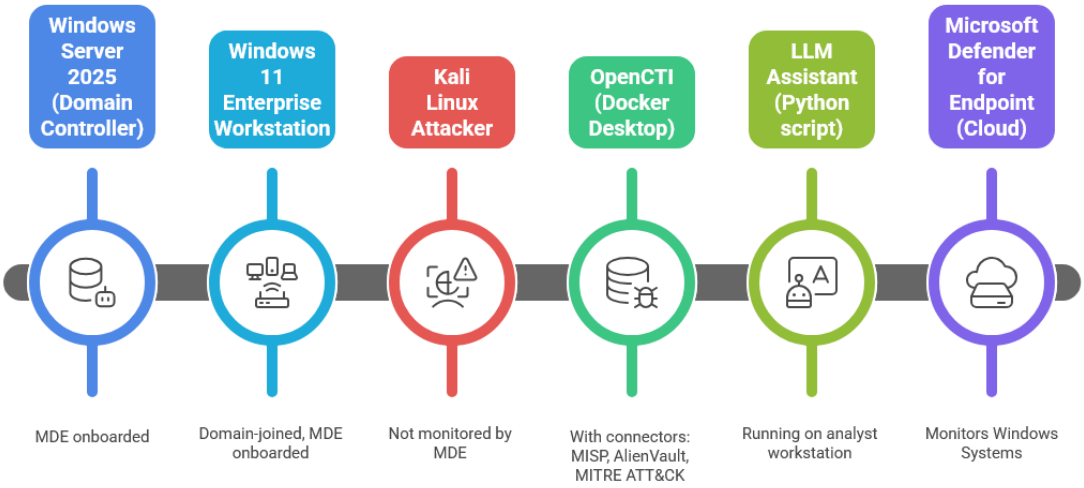

Figure 1. Experimental environment architecture.

 doi:10.20944/preprints202603.1994.v1

The system is deployed inside a multi-VM laboratory environment that emulates a small corporate network. All VMs run in Oracle VirtualBox (version 7.1.10) with identical networking configurations: each VM uses one NAT interface to access the Internet and one host-only adapter to communicate internally. This allows MDE to function as if deployed in an actual enterprise, while preserving isolation from the production network.

The environment consists of the following machines:

- Windows Server 2025 (OS build 26100.4946, August 12, 2025 update): Configured as an Active Directory Domain Services (AD DS) controller hosting the domain and authenticating enterprise users. A Global Administrator account is logged in to facilitate management operations.
- Windows 11 Enterprise (OS build 26100.4946, August 12, 2025 update): Domain-joined and used as the primary employee workstation. The domain user "Bob Worker" is logged in during experiments.
- Kali Linux (Rolling release June 13, 2025, kernel 6.12.0): Used exclusively as an attacker machine. This VM performs all offensive activities during the evaluation phase.

Both Windows VMs allocate 4 CPU cores and 6 GB of RAM, while the Kali machine uses 4 CPU cores and 4 GB RAM. Importantly, the Kali machine is not monitored by MDE, since it acts as the threat source. Additionally, on both Windows machines, local Microsoft Defender Antivirus and local Windows Firewall protections are fully disabled, ensuring that all detection and blocking events originate exclusively from the cloud-based MDE environment. This guarantees full visibility into the telemetry and prevents local security mechanisms from interfering with test results.

An overview of the laboratory environment components, their operating systems, resource allocations, and functional roles is provided in Table 2.

Table 2. Laboratory environment components and specifications.

Component	OS / Version	CPU	RAM	Role
Windows Server VM	Windows Server 2025 (OS build 26100.4946, Aug 12 2025 update)	4	6 GB	Active Directory Domain Controller
Windows 11 VM	Windows 11 Enterprise (OS build 26100.4946, Aug 12 2025 update)	4	6 GB	Domain-joined workstation
Kali VM	Kali Linux (Rolling, Jun 13 2025; kernel 6.12.0)	4	4 GB	Attacker's machine (not onboarded to MDE)

Figure 2 depicts the virtual network topology used in the laboratory environment, including the Windows Server domain controller, the Windows 11 workstation, and the Kali Linux attacker machine, along with their network connectivity.

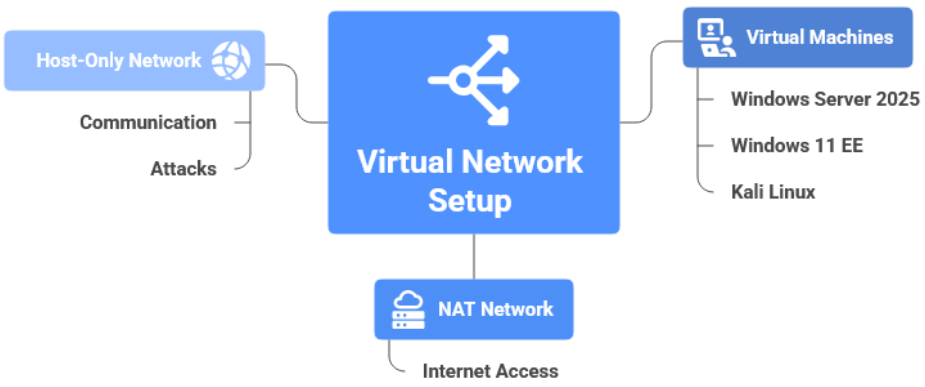

Figure 2. Virtual network topology overview.

 doi:10.20944/preprints202603.1994.v1

4.2. Deployment of MDE

The environment is connected to MDE with a P2 license, providing access to both advanced hunting and custom detection Rules. When onboarding the Windows VMs, the MDE agent begins forwarding all telemetry, including process creation, network connections, authentication events, file operations, and identity-related logs, to Microsoft 365 Defender. The principal value of MDE is twofold. First, it serves as the central telemetry collection system to validate the effectiveness of the generated KQL queries. Second, it provides the backend where generated detection rules are deployed and used to flag malicious behavior observed during subsequent attack simulations.

As already pointed out, the separation between local and cloud-based controls ensures that detections from the LLM-generated rules are genuine and not influenced by local antivirus heuristics. The MDE Advanced Hunting environment is used to test the syntactic correctness and logical precision of the generated queries before converting them into detection rules.

4.3. OpenCTI Threat-Intelligence Platform

The threat-intelligence backend is built on OpenCTI version 6.7.5, deployed on Docker Desktop 4.54.0. OpenCTI acts as a dynamic knowledge graph aggregating threat-intelligence artifacts from multiple connectors. In this implementation, three connectors feed the platform:

1. MITRE ATT&CK: Provides the entire matrix of tactics, techniques, sub-techniques, and associated metadata.
2. MISP-feed: Delivers indicators and event collections from publicly available MISP instances.
3. AlienVault OTX: Enriches OpenCTI with community-generated indicators, threat actors, campaigns, and observable relationships.

The Python assistant queries OpenCTI using GraphQL over HTTPS, authenticated with an API key. Although only MITRE ATT&CK objects are explicitly requested, their internal representation in OpenCTI may include cross-references, additional descriptions, and contextual relationships provided indirectly by the other connectors. Therefore, even though the assistant does not extract raw IoCs from MISP or OTX, the resulting MITRE technique descriptions may be richer than the canonical MITRE dataset.

OpenCTI's role in the architecture is to serve as a single structured intelligence source from which the LLM receives formal technique descriptions, platform information, kill chain phases and ATT&CK metadata, all of which shape the final KQL output.

Figure 3 shows the OpenCTI knowledge graph, illustrating how threat actors, attack techniques, observables, and related intelligence entities are interconnected within the platform.

 doi:10.20944/preprints202603.1994.v1

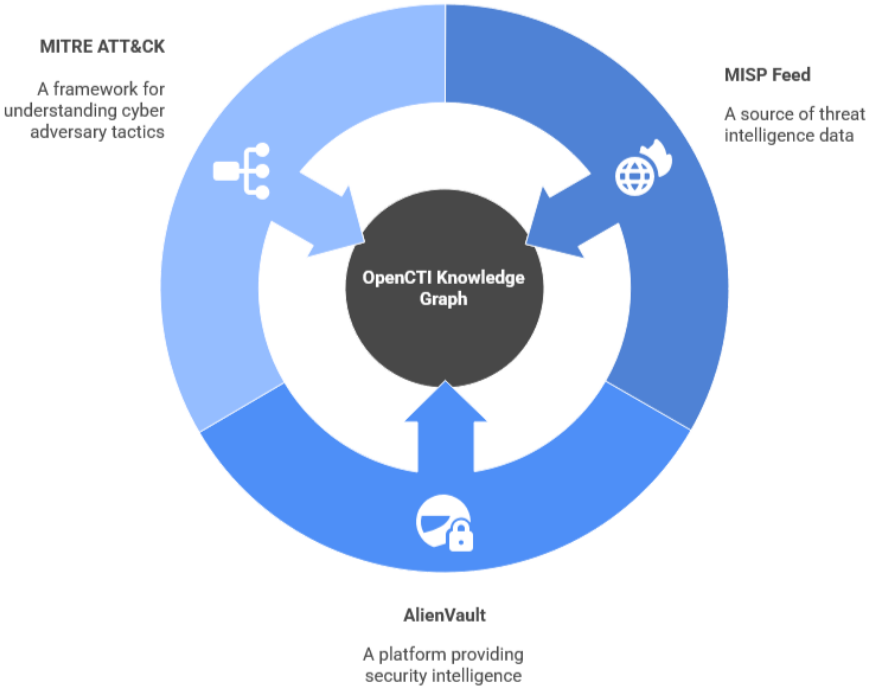

Figure 3. High-level schematic of the the OpenCTI knowledge graph.

4.4. Python-Based LLM Assistant Implementation

The LLM assistant is implemented in Python 3.13.5 using the OpenAI ChatGPT 5 (gpt-5-mini) model accessed through LangChain's ChatOpenAI wrapper. The assistant is executed manually from the Visual Studio Code terminal by a human analyst. Its operation follows a clearly defined workflow: first, the analyst inputs a MITRE technique by either its external ID (e.g., T1558.003) or its name (e.g., Kerberoasting). The assistant sends a GraphQL query to the OpenCTI backend searching for the relevant MITRE object. If found, OpenCTI returns its full metadata, including the technique name, description, kill chain phases, and platform applicability. The assistant extracts these fields and embeds them in a carefully engineered LLM prompt.

Second, the LLM receives a system message instructing it to behave as a senior SOC analyst who writes strictly correct and low-noise MDE KQL queries. The LLM is explicitly prohibited from producing explanatory text and must output only KQL. The prompt requests queries optimized to reduce false positives and false negatives, referencing only valid MDE schema fields and using appropriate event tables such as DeviceProcessEvents, DeviceNetworkEvents, DeviceLogonEvents and IdentityLogonEvents. Because LLMs occasionally hallucinate field names or use incorrect schema identifiers, the implementation incorporates a post-processing stage. This stage corrects known schema mismatches, especially around inconsistent uses of CommandLine fields across MDE event tables. Additionally, DeviceId and ReportId are automatically inserted into the projection stage to ensure query compatibility with MDE's Advanced Hunting environment.

The assistant then presents an initial KQL query to the analyst and enters an interactive refinement loop. The analyst may specify environment-specific tuning requirements such as accounts to exclude, business hours to consider, or device groups of interest. A revised prompt is generated incorporating these constraints, and the LLM produces a new query. This iterative procedure continues until the analyst approves the final KQL.

The assistant's prompt engineering is centered on a deterministic mapping logic that translates STIX 2.1 object attributes into specific MDE telemetry tables. For instance, when the OpenCTI metadata identifies a MITRE ATT&CK technique as network-based (e.g., T1021.002), the system message instructs the LLM to prioritize the DeviceNetworkEvents and DeviceLogonEvents tables. Conversely, for directory-service reconnaissance (e.g., T1087), the assistant guides the model toward Identity-

 doi:10.20944/preprints202603.1994.v1

DirectoryEvents or IdentityQueryEvents, ensuring that the generated KQL is not only syntactically correct but telemetry-aligned. This pre-filtering of schemas prevents the model from attempting to join unrelated high-volume tables, thereby reducing the computational cost of the resulting query.

4.5. System-Level Data Flow

The data flow begins with the analyst and ends with the deployment of a detection rule in MDE. The architecture follows a sequential interaction pattern where the LLM acts as the translation layer between threat-intelligence data and MDE detection logic.

A high-level diagram consists of four sequential blocks connected with directional arrows:

1. Analyst Input → MITRE technique ID or name is submitted through the Python script.
2. OpenCTI Retrieval → The script queries OpenCTI, retrieves MITRE metadata and extracts technique context.
3. LLM Processing → The extracted metadata is transformed into a structured prompt; the LLM outputs tuned KQL; post-processing ensures schema correctness.
4. MDE Integration → The analyst tests the query in Advanced Hunting, validates detections generated during simulated attacks, and converts it into a custom detection rule.

Figure 4 illustrates the OpenCTI knowledge ingestion pipeline, depicting how threat intelligence from multiple external sources is collected, normalized, and integrated into a unified knowledge base.

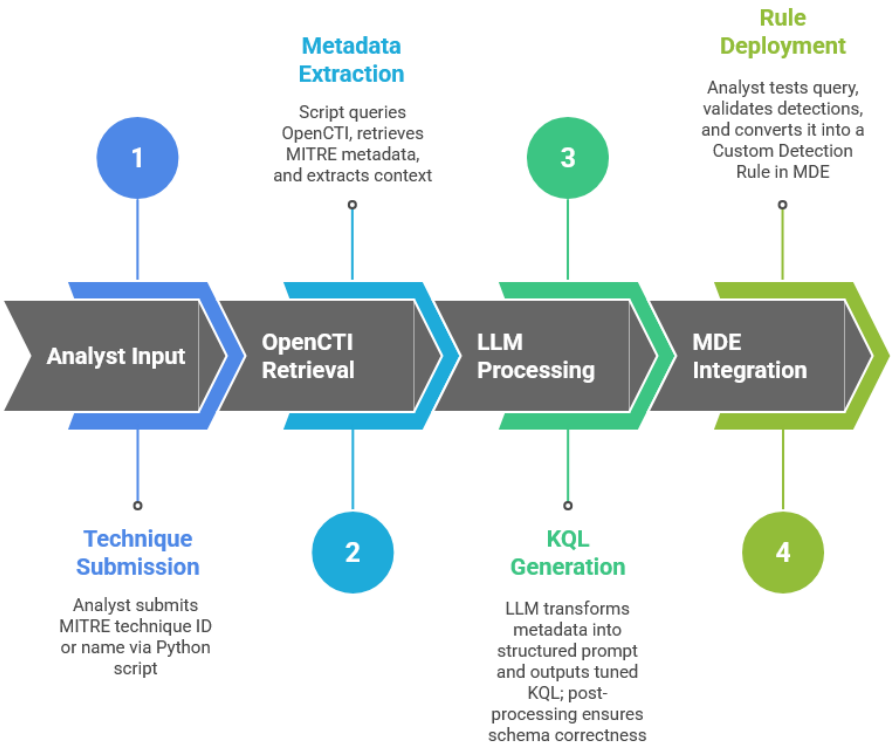

Figure 4. OpenCTI knowledge ingestion pipeline.

Figure 5 models the virtual environment. It consists of three VMs (Windows Server, Windows 11, Kali) placed within a rectangle representing VirtualBox. NAT and Host-Only networks are shown as two horizontal layers connecting all VMs. A cloud icon labeled "Microsoft Defender for Endpoint" is positioned outside the VirtualBox boundary, with arrows pointing from the Windows VMs to the cloud. The Kali VM has no arrow to MDE. OpenCTI and Docker Desktop are shown as an external intelligence node connected only to the Python assistant, not to the monitored machines.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 March 2026 doi:10.20944/preprints202603.1994.v1

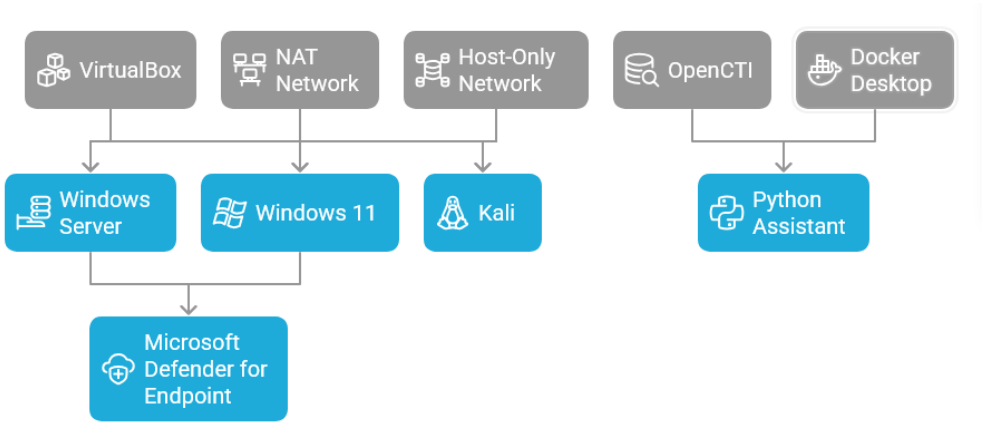

Figure 5. Virtual environment and security integration.

The logical integration of OpenCTI connectors is illustrated in Figure 6. MITRE ATT&CK, MISP-Feed, and AlienVault appear as three separate blocks feeding into an OpenCTI knowledge graph. The Python script queries only the MITRE subgraph, but the MITRE node is shown with dotted connections to other connectors to indicate indirect enrichment.

Overall, these three figures highlight that the LLM assistant does not interact with raw endpoint telemetry; instead, it bridges structured intelligence with MDE's detection stack.

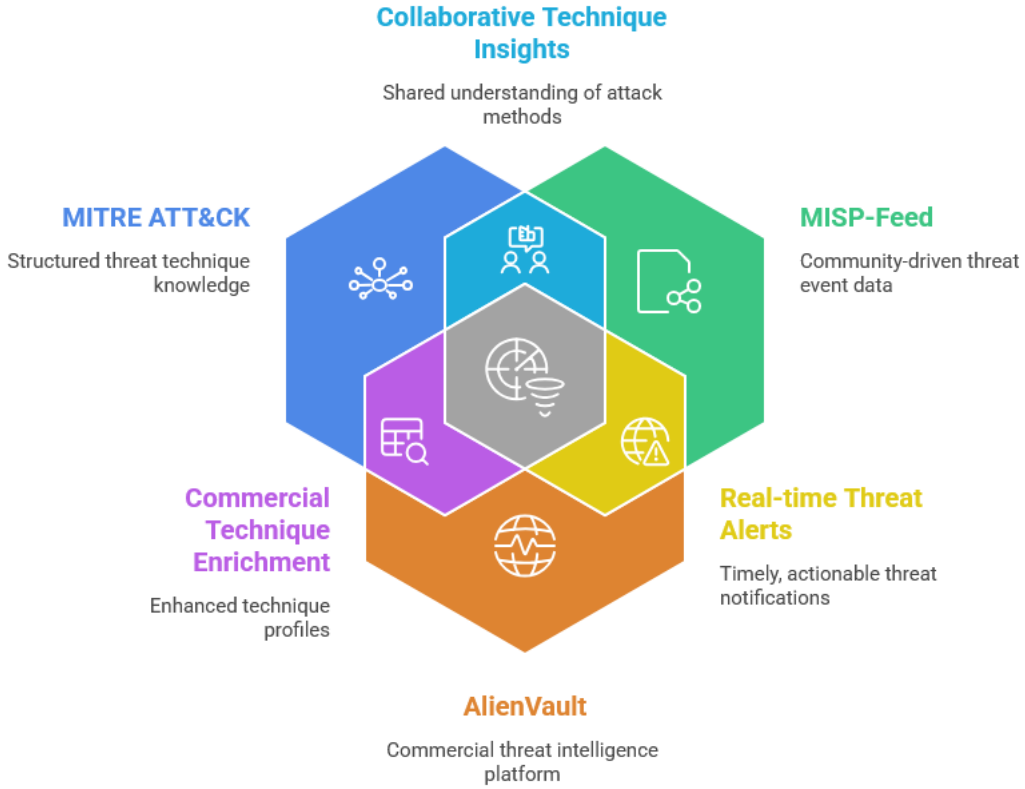

Figure 6. Unified threat intelligence enrichment.

4.6. Challenges

Several practical difficulties emerged during development. The first relates to inconsistent or invalid KQL generated by the LLM. Even with a restrictive prompt and explicit constraints, the model occasionally produced non-existent schema fields or mismatched device tables. This issue reflected the content variability of the model's training data. The post-processing module mitigated this by enforcing known field substitutions and guaranteeing mandatory fields. However, the LLM

 doi:10.20944/preprints202603.1994.v1

can sometimes still produce schematic errors, even when using a "reasoning-heavy" model such as o4-mini-deep-research.

A second key challenge arose from OpenCTI's enrichment behavior. Although the assistant queries only MITRE objects, the presence of multiple connectors means the MITRE description field may contain additional content. In certain instances, the LLM produced overly broad detections because the text provided by OpenCTI contained multiple threat references that were not directly related to the technique. This constraint was mitigated by truncating excessively long technique descriptions.

Finally, it was necessary to prevent local Windows Defender components from interfering with the evaluation. Their default behavior can block exploit attempts before MDE telemetry captures them. Disabling all local security features ensured that only MDE observations triggered alerts.

5. Evaluation

The evaluation phase constitutes the practical foundation for assessing the LLM-assisted detection engineering workflow. This section presents the deployment, configuration, and operational integration of the OpenCTI–LLM pipeline within the MDE-monitored infrastructure. It documents the execution of attack simulations across two distinct stages: a baseline phase (native MDE) and an augmented phase (custom LLM-generated rules).

5.1. Environment

The primary objective guiding the implementation was to create a reproducible, tightly controlled environment that accurately reflects realistic enterprise conditions while simultaneously allowing repeatable offensive testing. The architectural approach was intentionally minimalist, avoiding unnecessary complexity or confounding variables. This allowed the evaluation to focus specifically on the interplay between (a) MDE's native heuristics, (b) the LLM-assisted rule-generation pipeline, and (c) adversarial techniques aligned with the MITRE ATT&CK framework.

As detailed in Section 4.1, the environment consisted of monitored Windows hosts and an isolated Kali Linux attacker machine. The exclusion of the attacker system from MDE monitoring was pivotal to ensure that telemetry reflected only the impact on targets rather than the presence of offensive tools on the source. Software versions were selected based on contemporaneity, utilizing August 2025 builds to ensure the relevance of the findings (see Table 3).

Table 3. Software Stack and Versions.

Software / Platform	Version / Notes
Windows Server	Windows Server 2025 (OS build 26100.4946, August 12, 2025 update)
Windows 11	Windows 11 Enterprise (OS build 26100.4946, August 12, 2025 update)
Kali Linux	Rolling release (June 13, 2025), kernel 6.12.0
OpenCTI	6.7.5
Docker Desktop	4.54.0
Python	3.13.5
Oracle VirtualBox	7.1.10
Microsoft Defender for Endpoint	P2 license

As already mentioned in Section 4.1, the monitored Windows hosts were deployed with symmetrical resource allocations (4 CPU cores, 6 GB RAM). A dual-adapter network topology – NAT for cloud-based telemetry offloading and Host-only for internal lateral movement – ensured that the attacker could reach targets within a closed intranet while MDE maintained its command-and-control (C2) connection to the Microsoft 365 Defender backend. The Active Directory configuration supported realistic SMB, WMI, Kerberos, and LDAP traffic patterns, providing a representative telemetry surface.

 doi:10.20944/preprints202603.1994.v1

5.2. Intelligence Integration and LLM Pipeline

With reference to Section 4.3, OpenCTI (v6.7.5) was deployed via Docker, integrating three primary intelligence streams: MISP, AlienVault OTX, and the MITRE ATT&CK connector. Although the LLM relies on MITRE ATT&CK v18.0 (which emphasizes behavioral analytics), enrichment from auxiliary feeds provided the operational context, e.g., command-line patterns and network indicators, necessary to improve the precision of the generated KQL, particularly for stealthy techniques like credential dumping.

Recall from Section 4.4 that the Python-based assistant orchestrates the pipeline by fetching STIX metadata from OpenCTI and synthesizing it into MDE KQL using the GPT-5-mini model. A critical feature is the post-processing engine, which performs schema normalization. Given that MDE's schema is subject to frequent updates, the script automatically corrects deprecated field names in tables such as DeviceProcessEvents and DeviceNetworkEvents, while ensuring the injection of mandatory DeviceId and ReportId fields.

The analyst initiates the rule-generation process manually by entering a MITRE technique identifier or technique name into the script. The LLM then retrieves the relevant technique description, associated kill-chain phases, supported platforms, and a condensed version of the technique summary from OpenCTI. Based on this information, it constructs a custom prompt with operational constraints emphasizing noise reduction, correct schema usage, and inline comments for further tuning. The script then enters an interactive refinement loop that allows an analyst to introduce environment-specific filters such as excluded devices, benign service accounts, business-hours constraints, or high-value device groups. This iterative tuning is critical because a detection rule tailored to a generic environment is typically noisy and prone to false positives, whereas rules aligned with real-world operational context tend to be significantly more stable.

In the evaluation, however, initial rule testing was conducted without applying the refinement loop, in order to measure the baseline performance of the LLM's raw output. Only syntactic errors were corrected where necessary to allow execution in MDE's Advanced Hunting interface.

5.3. Execution of Attacks Prior to Custom Rule Deployment

Before introducing LLM-generated rules, each attack scenario given in Table 1 was executed to establish the out-of-the-box visibility of MDE.

- Phishing (T1566): Implemented via HiddenEye to harvest credentials. As this relied on user interaction rather than malicious process execution, MDE failed to generate alerts, highlighting a significant gap in endpoint-only visibility.
- Account Discovery (T1087): Executed using the netexec framework. Despite the generation of authentication logs, MDE yielded zero alerts, confirming that reconnaissance often remains below the detection threshold of native heuristics.
- Kerberoasting (T1558.003): Executed via Atomic Red Team. This was reliably detected and blocked by MDE's native engine, generating high-severity incidents.
- SMB Reconnaissance (T1135/T1021.002): While MDE flagged the subsequent reverse-shell delivery, the initial share enumeration was inconsistently detected, indicating a lack of early-stage visibility.
- Credential Dumping (T1003): While secretsdump.py triggered alerts, newer tools like regsecrets.py were silently blocked without generating an incident, demonstrating a prevention-detection mismatch where the analyst is left unaware of the thwarted attempt.

5.4. Evaluation of LLM-Generated Rule Behavior

After establishing baseline behavior, all attack scenarios were re-executed with LLM-generated detection rules deployed in MDE. Each rule was first validated as an Advanced Hunting query to ensure schema correctness and was subsequently converted into a custom detection rule. Due to

 doi:10.20944/preprints202603.1994.v1

platform limitations, only the account discovery rule was configured to run in near–real-time (NRT) mode, as most custom analytics in MDE are restricted to scheduled execution.

The LLM-generated phishing detection rule was adapted slightly to match the specific characteristics of the locally hosted HiddenEye phishing page. Once deployed, the rule successfully generated alerts when credentials were submitted through the fake login page. However, due to the inability of custom rules to operate in NRT mode and limitations in MDE response actions, real-time blocking and automatic remediation were not feasible. This outcome confirms that phishing detection at the endpoint level is possible through behavioral analytics but remains constrained to alerting rather than prevention.

When the LLM-generated account discovery rule was deployed in NRT mode, user enumeration activity that previously went undetected was successfully identified. The rule generated alerts and incidents shortly after execution, providing immediate analyst visibility. This represents one of the most significant improvements observed in the study, demonstrating that LLM-assisted detection engineering can effectively address blind spots in native MDE heuristics related to reconnaissance activity.

The effectiveness of these rules is rooted in the interactive refinement loop, which allows the analyst to move from a broad behavioral detection to a hardened operational rule. For example, an initial LLM-generated query for account discovery might flag all net.exe executions. During the refinement process, the analyst can specify that legitimate administrative service accounts should be excluded. The LLM then autonomously updates the KQL to include a `where AccountName !in (“svc_admin”, “backup_user”)` filter and adds logic to calculate a threshold (e.g., *summarize count() by DeviceId | where count > 10)*, effectively suppressing the noise typical of standard domain operations while maintaining high-fidelity visibility into adversarial enumeration.

Kerberoasting attacks were re-executed using the same methodology as in the baseline phase. MDE continued to detect and block the activity reliably. The LLM-generated rule did not trigger in this scenario, likely due to differences in how IoC and behavioral patterns were represented in the OpenCTI dataset. This result suggests that for techniques with strong native detection coverage, LLM-generated rules may provide limited additional value.

Following the deployment of the LLM-generated rule, SMB share enumeration activity that previously went unnoticed was successfully detected and alerted on. Reverse-shell activity continued to be detected by MDE’s native analytics. This demonstrates that LLM-generated detection logic can complement existing MDE detections by extending coverage to earlier reconnaissance stages of lateral movement.

For secretsdump.py, the LLM-generated detection rule triggered successfully, although native MDE detection already provided adequate coverage. More importantly, when credential dumping was performed using the LSASSY module in netexec, the LLM-generated rule raised alerts where MDE had previously produced only a silent block.

In contrast, pass-the-hash–related rules exhibited mixed results. The generated query required manual correction due to schema inconsistencies and invalid field references, preventing immediate deployment. This highlighted recurring limitations in the LLM’s understanding of the MDE schema, even when post-processing was applied.

The most notable result of the evaluation was the complete absence of native MDE detection for account discovery and SMB share enumeration, contrasted with successful alert generation by the LLM-generated rules operating in NRT mode. These findings provide strong evidence that the proposed LLM-enhanced threat intelligence pipeline meaningfully supplements MDE by addressing detection gaps in discovery and early-stage lateral movement techniques [27].

Figure 7 presents the classification scheme used to categorize detection outcomes observed during the experimental evaluation.

 doi:10.20944/preprints202603.1994.v1

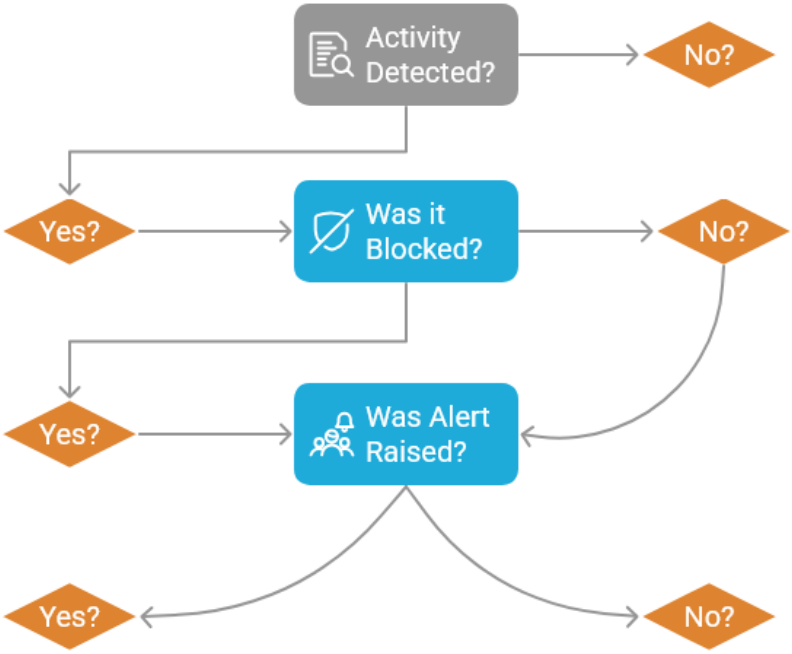

Figure 7. Detection outcome classification.

6. Discussion

In this section, we synthesize the empirical findings to examine the broader implications of integrating an LLM–driven assistant into an enterprise detection pipeline centered on MDE. The analysis is structured around the detection efficacy of out-of-the-box MDE, the operational value of LLM-generated KQL, and the comparative performance metrics that define the current boundaries of EDR/XDR architectures. Through this synthesis, we highlight how LLM-assisted detection engineering serves as a critical bridge between raw threat intelligence and hardened security controls.

6.1. Native MDE Detection Performance: The Prevention-Visibility Paradox

The initial baseline phase, which executed all attack scenarios without custom detections, established that MDE possesses sophisticated robustness against high-signal threats such as Kerberoasting, traditional credential dumping via secretsdump.py, and reverse-shell creation. In these instances, the platform's heuristic engines reliably triggered alerts and automated remediation. The detection of Kerberoasting is particularly significant, as Kerberos ticket abuse is notoriously difficult to disambiguate from legitimate administrative workflows.

However, the evaluation of regsecrets.py and LSASSY revealed a systemic gap we characterize as the "prevention-visibility paradox". In these scenarios, MDE performed a silent block, successfully thwarting the exploit but failing to generate an incident or alert for the SOC. While the immediate compromise is prevented, the absence of a forensic trail means that analysts cannot determine the identity, intent, or origin of the actor. This lack of situational awareness allows attackers to probe system defenses iteratively without triggering a response, effectively granting the adversary a low-risk environment for discovery. This behavior underscores a shift in modern EDR design where prevention is prioritized over telemetry, a gap that our LLM-generated logic successfully bridged by restoring visibility to these thwarted attempts.

The observation that regsecrets.py and LSASSY were silently blocked suggests a disconnection between the EDR's enforcement and reporting layers. We hypothesize that these attempts triggered attack surface reduction rules or kernel-level protections that thwarted the process injection before the user-mode telemetry sensors could generate a standardized alert object. While the immediate threat is neutralized, this visibility gap is critical. Without the custom LLM-generated rules to capture the raw

 doi:10.20944/preprints202603.1994.v1

DeviceEvents associated with the failed attempt, the SOC remains blind to the adversary's presence, allowing them to refine their exploitation strategy undetected.

6.2. Operational Efficacy of LLM-Generated KQL

After establishing the baseline, the re-execution of attacks using LLM-generated rules demonstrated that the assistant could produce queries semantically aligned with technique descriptions and structured around the appropriate MDE schema tables. The LLM significantly augmented visibility in domains where MDE was previously blind, most notably in account discovery and user enumeration. By deploying these rules in NRT mode, we transformed silent reconnaissance activity into actionable alerts, providing the SOC with the early-warning indicators necessary to disrupt an attack before it reaches the lateral movement phase.

Despite these successes, the practical implementation revealed that the semantic gap between public threat intelligence and proprietary schemas remains a persistent hurdle. Even with reasoning-heavy models, the assistant occasionally proposed deprecated field names or logic that assumed the existence of telemetry that MDE's logging filters may have suppressed. This highlights that while LLMs excel at logical synthesis, they require a robust post-processing layer to ensure that the generated KQL is syntactically compatible with the ever-evolving MDE schema. Furthermore, the assistant's performance in detecting Phishing showed that while behavioral analytics can identify credential submission, the latent nature of non-NRT rules remains a constraint for real-time interception.

6.3. Comparative Assessment and the Intelligence-to-Logic Latency

As summarized in Table 4, the comparative assessment confirms that LLM-driven detection engineering acts as a complementary extension rather than a replacement for native EDR heuristics. We observe a clear divergence in coverage: native MDE remains the authoritative source for complex, identity-centric attacks like Kerberoasting, whereas LLM-derived rules provide the granular visibility required for reconnaissance and SMB-based discovery.

A primary takeaway of this study is the reduction in intelligence-to-logic latency. Traditionally, transforming a new MITRE technique into a deployed KQL hunter requires manual extraction, research, and testing by a senior engineer, a process that can take hours or days. Our implementation demonstrates that an LLM, grounded in OpenCTI metadata, can reduce this lifecycle to minutes. This acceleration is vital for defending against agile APT groups that frequently rotate their TTPs to bypass static detection patterns. By automating the resolution of MITRE techniques and the synthesis of KQL, the assistant allows the SOC to maintain a dynamic defensive posture that evolves at the speed of the threat landscape.

That is, beyond detection efficacy, the study observed a drastic reduction in the intelligence-to-logic lifecycle. While the manual authoring of a high-fidelity KQL query for a technique like Kerberoasting typically requires 30 to 45 min of schema cross-referencing and syntax testing by a senior engineer, the LLM-powered assistant synthesized executable queries in sub-60-sec intervals. When factoring in the automated OpenCTI retrieval, the entire process – from threat discovery to a deployed custom detection – was reduced from hours to under five min, representing a significant operational force multiplier for resource-constrained security teams.

 doi:10.20944/preprints202603.1994.v1

Table 4. Detection results summary.

Attack Technique	Native MDE Detection	Native MDE Blocking	Custom Rule Detection	Rule Execution Mode (NRT / Scheduled)
Phishing (HiddenEye)	No	No	Yes (after tuning)	Scheduled
Account Discovery / User Enumeration	No	No	Yes	NRT
Kerberoasting	Yes	Yes	No (baseline)	Scheduled
SMB share enumeration / SMB-based activity	Mixed	Mixed	Yes (enumeration)	Scheduled
Credential dumping (secretsdump.py)	Yes	Yes	Yes	Scheduled
Credential dumping (regsecrets.py / LSASSY)	No (silent)	Yes (silent)	Yes	Scheduled
Pass-the-hash	Mixed	Mixed	Mixed (schema fixes needed)	Scheduled

6.4. Limitations and Operational Impact

Several limitations are intrinsic to the proposed architecture, the most significant being the dependency on underlying telemetry. KQL detections, regardless of their logical sophistication, cannot exist if the host-level sensors do not capture the event. For instance, certain SMB-based activities occur within benign administrative noise, making them indistinguishable from legitimate traffic without deep-packet inspection, which MDE may not expose to the Advanced Hunting interface.

Additionally, one should acknowledge the schema hallucination risk. The proprietary nature of MDE's telemetry means the LLM is often reasoning based on stale or generalized training data. This necessitates a human-in-the-loop model where the analyst performs the final validation and tuning, ensuring the rule is optimized for the specific noise profile of the enterprise environment. Finally, MDE's throttling of NRT rules imposes a structural limit on how many LLM-generated detections can be preemptive. Most custom analytics remain reactive, functioning more as automated hunting tools than real-time prevention mechanisms.

The integration of MDE, OpenCTI, and an LLM-powered assistant realizes a hybrid detection strategy that leverages the unique strengths of each component. MDE provides the foundation of behavioral prevention for high-risk exploits, while the LLM facilitates the creation of intelligence-driven detections for low-signal reconnaissance. OpenCTI serves as the connective tissue, providing the structured context that prevents the LLM from operating in a vacuum. This synergy eliminates silent failure scenarios, restores forensic context, and empowers analysts to move from a reactive state to a proactive, threat-informed defensive posture.

This experimental evaluation underscores a significant shift in defensive capabilities when EDR platforms are augmented by AI-driven logic. While native MDE remains the authoritative barrier against high-signal exploits like Kerberos-based attacks, the integration of an LLM-powered assistant effectively closes the visibility gaps inherent in reconnaissance and early-stage lateral movement. By bridging the prevention-visibility paradox through automated KQL synthesis and OpenCTI enrichment, the proposed framework demonstrates that a hybrid strategy, combining heuristic-based prevention with intelligence-driven custom analytics, provides a more resilient and transparent defensive posture. Ultimately, this approach scales the SOC's ability to respond to agile adversaries by reducing the operational latency between threat discovery and deployed logic.

 doi:10.20944/preprints202603.1994.v1

7. Conclusions and Future Work

The primary objective of this work was to evaluate whether LLMs can materially enhance the threat detection capabilities of MDE by transforming structured intelligence from OpenCTI into MITRE-aligned KQL rules. The findings confirm that such an integration is not only feasible but provides a significant operational force multiplier, reducing the intelligence-to-logic latency from hours to under five minutes. By simulating adversarial activity in a high-fidelity laboratory, this study has demonstrated that AI-assisted detection engineering effectively addresses structural blind spots in native EDR heuristics, particularly within the discovery and reconnaissance phases.

The findings of this research successfully address the core research questions of Section 1 by demonstrating that LLMs can accurately contextualize graph-based intelligence (RQ1/RQ2) and transform it into syntactically valid KQL (RQ3). No less important, the empirical results confirm that this automation significantly reduces analyst workload (RQ4) by shrinking the intelligence-to-logic latency from hours to minutes, while simultaneously filling critical visibility gaps in native EDR heuristics.

Simultaneously, the experiments revealed critical technical constraints. The tendency for LLM outputs to occasionally produce non-existent schema fields necessitates a robust post-processing layer, confirming that fully autonomous rule generation is not yet a viable replacement for expert oversight. Furthermore, the architectural model of MDE, which favors scheduled over NRT execution for custom rules, somewhat tempers the immediate operational impact of certain generated detections.

The findings suggest that LLMs significantly augment detection engineering by accelerating the rule-creation lifecycle and expanding coverage for under-instrumented, frequently-mutated adversarial behaviors. However, these models cannot circumvent the telemetry limitations inherent to EDR platforms. Their primary value lies in streamlining the translation of raw intelligence into actionable logic and facilitating the development of detections where native behavioral models are absent.

Altogether, LLM-assisted detection engineering represents a transformative complementary capability for modern security operations. When integrated with structured platforms like OpenCTI, it scales detection coverage and enhances the agility of defensive teams, provided that human expertise remains at the center of the validation and deployment process.

Future work should explore closed-loop automated IoC-to-rule correlation pipelines and multi-agent LLM-based rule validation mechanisms to reduce the need for manual post-processing. Additionally, evaluations in larger, production-scale enterprise environments are required to assess how LLM-generated rules perform under high alert volumes and complex noise profiles. Further research may also examine cross-platform detection synthesis to provide a unified defensive language across disparate SIEM and EDR ecosystems.

Author Contributions: Conceptualization, I.K., G.K, E.C.; methodology, I.K., E.C.; writing—original draft preparation, I.K, K.K., G.K.; writing—review and editing, K.K., G.K.; visualization, I.K.; supervision, G.K, E.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

Data Availability Statement: The Python-based prototype used in the context of this work is made available at a public GitHub repository [25].

References

1. Kampourakis, K.E.; Gkioulos, V.; Kavallieratos, G.; Lin, J.C. Digital Twin-Enabled Incident Detection and Response: A Systematic Review of Critical Infrastructures Applications. *International Journal of Information Security* **2025**, *24*, 194. https://doi.org/10.1007/s10207-025-01113-0.
2. Arfeen, A.; Ahmed, S.; Khan, M.A.; Jafri, S.F.A. Endpoint Detection & Response: A Malware Identification Solution. In Proceedings of the 2021 International Conference on Cyber Warfare and Security (ICCWS), 2021, pp. 1–8. https://doi.org/10.1109/ICCWS53234.2021.9703010.

 doi:10.20944/preprints202603.1994.v1

3. Alturkistani, H.; Chuprat, S. Artificial intelligence and large language models in advancing cyber threat intelligence: A systematic literature review **2024**. https://doi.org/https://doi.org/10.21203/rs.3.rs-5423193/v1.
4. Kampourakis, K.E.; Kampourakis, V.; Chatzoglou, E.; Kambourakis, G.; Gritzalis, S. Knowledge-to-Data: LLM-Driven Synthesis of Structured Network Traffic for Testbed-Free IDS Evaluation. *arXiv preprint arXiv:2601.05022* **2026**. https://doi.org/https://doi.org/10.48550/arXiv.2601.05022.
5. Smiliotopoulos, C.; Kambourakis, G.; Kolias, C. Detecting Lateral Movement: A Systematic Survey. *Heliyon* **2024**, *10*, e26317. https://doi.org/10.1016/j.heliyon.2024.e26317.
6. Ho, G.; Dhiman, M.; Akhawe, D.; Paxson, V.; Savage, S.; Voelker, G.M.; Wagner, D. Hopper: Modeling and detecting lateral movement. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 3093–3110.
7. Hassan, W.U.; Bates, A.; Marino, D. Tactical provenance analysis for endpoint detection and response systems. In Proceedings of the 2020 IEEE symposium on security and privacy (SP). IEEE, 2020, pp. 1172–1189. https://doi.org/10.1109/SP40000.2020.00096.
8. Kaur, Harpreet.; Sanjaiy SL, Dharani.; Paul, Tirtharaj.; Kumar Thakur, Rohit.; Kumar Reddy, K. Vijay.; Mahato, Jay.; Naveen, Kaviti. Evolution of Endpoint Detection and Response (EDR) in Cyber Security: A Comprehensive Review. *E3S Web Conf.* **2024**, *556*, 01006. https://doi.org/10.1051/e3sconf/202455601006.
9. Yusof, Z.B. Effectiveness of endpoint detection and response solutions in combating modern cyber threats. *Journal of Advances in Cybersecurity Science, Threat Intelligence, and Countermeasures* **2024**, *8*, 1–9.
10. Al-Sada, B.; Sadighian, A.; Oligeri, G. MITRE ATT&CK: State of the art and way forward. *ACM Computing Surveys* **2024**, *57*, 1–37. https://doi.org/https://doi.org/10.1145/368730.
11. Rajesh, P.; Alam, M.; Tahernezhadi, M.; Monika, A.; Chanakya, G. Analysis of cyber threat detection and emulation using mitre attack framework. In Proceedings of the 2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA). IEEE, 2022, pp. 4–12. https://doi.org/10.1109/IDSTA55301.2022.9923170.
12. Xu, H.; Wang, S.; Li, N.; Wang, K.; Zhao, Y.; Chen, K.; Yu, T.; Liu, Y.; Wang, H. Large language models for cyber security: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* **2024**. https://doi.org/https://doi.org/10.1145/3769676.
13. Zhang, J.; Bu, H.; Wen, H.; Liu, Y.; Fei, H.; Xi, R.; Li, L.; Yang, Y.; Zhu, H.; Meng, D. When llms meet cybersecurity: A systematic literature review. *Cybersecurity* **2025**, *8*, 55. https://doi.org/https://doi.org/10.1186/s42400-025-00361-w.
14. Saura, P.F.; Jayaram, K.R.; Isahagian, V.; Bernabé, J.B.; Skarmeta, A. On Automating Security Policies with Contemporary LLMs, 2025, [arXiv:cs.CR/2506.04838].
15. Wang, H.; Xu, M.; Guo, Y.; Han, W.; Lim, H.W.; Dong, J.S. RulePilot: An LLM-Powered Agent for Security Rule Generation, 2025, [arXiv:cs.CR/2511.12224].
16. Anvilogic.; Institute, S. State of Detection Engineering Report. Technical report, Anvilogic, 2025.
17. OpenCTI Platform. OpenCTI Official Documentation. https://www.opencti.io/en/, 2024. Accessed: 2025-01.
18. OpenCTI Platform. OpenCTI Platform GitHub Repository. https://github.com/OpenCTI-Platform/opencti, 2024. Accessed: 2025-01.
19. MISP Project. MISP Threat Sharing Platform. https://github.com/MISP/MISP, 2024. Accessed: 2025-01.
20. Yeti Platform. Yeti Threat Intelligence Platform. https://github.com/yeti-platform/yeti, 2023. Accessed: 2025-01.
21. Prelude Security. Prelude Security. https://www.preludesecurity.com/, 2026. Accessed: 2026-01.
22. Silas Cutler. MalPipe: Malware/IOC ingestion and processing engine. https://github.com/silascutler/MalPipe, 2024. Accessed: 2026-01.
23. Tian, S.; Zhang, T.; Liu, J.; Wang, J.; Wu, X.; Zhu, X.; Zhang, R.; Zhang, W.; Yuan, Z.; Mao, S.; et al. Exploring the Role of Large Language Models in Cybersecurity: A Systematic Survey, 2025, [arXiv:cs.CR/2504.15622].
24. NIST. NIST AI Risk Management Framework. https://www.nist.gov/itl/ai-risk-management-framework, 2023. Accessed: 2025-01.
25. Johnnykonst. LLM-Assisted MITRE-Aware KQL Generation for Microsoft Defender for Endpoint. https://github.com/Johnnykonst/LLM-Assistant-for-Microsoft-Defender-for-Endpoint, 2026. Accessed: 2026-02.
26. Microsoft. Advanced Hunting in Microsoft Defender for Endpoint. https://learn.microsoft.com/en-us/defender-endpoint/advanced-hunting-overview, 2024. Accessed: 2025-01.

 doi:10.20944/preprints202603.1994.v1

27. Smiliotopoulos, C.; Barmpatsalou , K.; Kambourakis, G. Revisiting the Detection of Lateral Movement through Sysmon. *Applied Sciences* **2022**, *12*. https://doi.org/10.3390/app12157746.