Toward Human-Aware Active Directory Defense with Fine-Tuned LLMs

Hung X. Nguyen and Tu Vu

School of Computer and Information Technology
Adelaide University, Australia
first.last@adelaide.edu.au

Abstract

Autonomous cyber defense agents increasingly need to query complex enterprise attack graphs under time pressure, yet today's Active Directory (AD) tools still assume expert-authored Cypher queries. This paper studies natural-language-to-Cypher generation as a core tool-use action for human-aware AD defense agents operating over BloodHound-style graphs. We curate 346 executable English–Cypher pairs from practitioner queries, augment them to 2,768 samples via constrained paraphrasing, and fine-tune an open-weight Mixtral-8×7B model using QLoRA. Across ten representative BloodHound-equivalency tasks on five synthetic AD graphs, fine-tuning raises parse success from 0.80 to 0.94 and correct-answer rate from 0.34 to 0.42. These gains should be interpreted as evidence that domain tuning can improve syntactic robustness in a controlled setting, rather than as evidence of production-ready semantic reliability or real-enterprise scalability. Error analysis reveals safety-critical semantic gaps in domain admin identification, unsupported-OS detection, and temporal constraints. We argue that, for cyber defense agents, schema prompting and fine-tuning alone are insufficient: safe autonomy over AD graphs requires execution-grounded correction, explicit domain guardrails, and interaction designs that calibrate human trust in LLM-generated queries.

Introduction

Active Directory (AD) underpins authentication and authorization in a large fraction of enterprise Windows environments, making identity relationships a high-value target for adversaries and a high-priority surface for defenders (Dunagan, Zheng, and Simon 2009; Goel et al. 2022; Zhang et al. 2023; Guo et al. 2023; Ngo, Guo, and Nguyen 2024). AD compromise is rarely a single-step event: attackers chain delegated privileges, group nesting, and session relationships to escalate toward high-value principals such as Domain Admins. BloodHound operationalizes this reality by extracting AD security relationships into a graph, enabling analysts to reason about attack paths at scale (SpecterOps 2025).

The emerging "agentic" paradigm in cyber defense emphasizes systems that can rapidly interrogate complex state, use tools, and produce actionable outputs under time pressure—while remaining transparent and accountable to human operators. In this setting, graph queries are not merely a usability feature; they are a core *tool-use action* that defensive agents and copilots must perform reliably when triaging identity exposure and privilege escalation risk. However, BloodHound/-Neo4j analysis typically requires writing specialized Cypher queries over large, schema-rich graphs - a significant barrier for teams without both deep AD and graph-query expertise.

Large language models (LLMs) offer a natural-language interface for expressing investigative intent, but reliability is the central challenge in safety-critical workflows. As emphasized by operational evaluation efforts such as OCCULT (Kouremetis et al. 2025), models may produce fluent outputs that appear plausible yet fail in ways that would mislead analysts or amplify risk. For AD graph analytics, failures occur at two levels: (i) *syntactic* errors that prevent execution, and (ii) *semantic* errors where syntactically valid Cypher executes but retrieves the wrong result set (e.g., missing critical attack paths or producing spurious findings). These semantic failures are especially dangerous when an analyst or an autonomous agent over-trusts a seemingly coherent query. Recent practitioner systems (e.g., BloodHound MCP) illustrate a tool-centric approach in which an LLM is grounded via curated query patterns and iterative refinement using execution feedback (Nickerson 2025b,a). While effective, the resulting workflow can be operationally slow and heavily dependent on human-maintained exemplars.

This work is a preliminary study of LLM-mediated attack graph querying and asks: to what extent can domain-specific fine-tuning improve the reliability of this action for human-supervised AD defense workflows? This direction is attractive for autonomous defense settings where latency, auditability, and dependence on proprietary inference services can be limiting factors. Recent progress on natural-language interfaces for structured querying (Hristidis, Gravano, and Papakonstantinou 2003; Li and Jagadish 2014) and text-to-Cypher (Francis 2018; Ozsoy et al. 2025; Tiwari et al. 2025; Munir and Aldini 2025) highlights both the promise and brittleness of LLM-based query generation.

We present an LLM-enabled, tool-grounded pipeline that converts natural-language AD defense questions into executable Cypher actions over a BloodHound-style Neo4j attack graph. We make three contributions in this work: (1) A domain-specific **AD Text→Cypher** corpus of 346 executable English–Cypher pairs for BloodHound-style tool-use actions by cyber defense agents, augmented to 2,768 via controlled

paraphrasing; (2) Evidence that QLoRA fine-tuned Mixtral can substantially improve the syntactic reliability of LLM-generated graph queries in AD-focused agent pipelines; and (3) A preliminary error taxonomy showing why human-aware guardrails and execution-grounded correction are necessary before delegating AD querying to autonomous agents.

Our central position is that while fine-tuning largely eliminates syntactic failures, semantic reliability remains insufficient for unmonitored autonomy and must be complemented with execution-grounded and human-aware guardrails. All code and data is available at https://github.com/AUCyberLab/llm-hound.

System Overview

Figure 1 summarizes the full pipeline of our fine-tuning solution. Our workflow follows a "data → graph database → LLM → execution → user" pattern as detailed below.

AD Graph Generation with `DBCreator`

To evaluate our approach in a controlled setting and avoid using sensitive enterprise directories, we use synthetic AD graphs generated by BloodHound's official `DBCreator` (BloodHoundAD 2023). DBCreator data approximate real enterprise AD structures (e.g., numbers of users, groups, computers, group nesting depth, administrative assignments), producing graphs that contain both common and security-relevant relationships.

AD Text-to-Cypher Corpus

A central challenge in training LLMs for text-to-Cypher is the scarcity of high-quality, domain-aligned pairs of natural-language questions and executable Cypher queries. To address this, we construct an AD-specific corpus grounded in BloodHound-style querying over Neo4j property graphs. Each sample in the corpus is a pair ⟨English prompt, Cypher query⟩ tailored to AD entities and relationships, enabling supervised fine-tuning and execution-grounded evaluation.

We curate Cypher queries from a mix of official and practitioner-oriented AD security resources. The core of the corpus is the official SpecterOps BloodHound Query Library (GitHub), which we augment with practitioner 5 cheat sheets and blogs from AD researchers. We further incorporate 3 community-maintained GitHub resources and repositories. Together, these sources capture both canonical BloodHound patterns and diverse real-world queries.

Selection criteria. To ensure that the resulting dataset teaches both Cypher syntax and AD-specific semantics, we apply three selection criteria. First, we require *coverage of AD entities and relationships*. Queries must span common node types such as `User`, `Group`, and `Computer`, and include security-relevant relationships such as group membership/nesting, sessions, and administrative privileges. Second, we enforce *diversity in query complexity* by selecting prompts at multiple difficulty levels (from simple enumeration to multi-hop traversals and privilege-oriented queries) as shown in (Kouremetis et al. 2025).

We validated every candidate query by running it through the Neo4j Python driver. Queries that produce syntax errors or fail to execute are either repaired (when the intended semantics are clear) or removed.

Final curated corpus. After removing duplicates (based on near-identical English prompts or Cypher statements) and enforcing the above criteria, the final curated corpus contains 346 unique samples. Each sample is stored in JSON format with three fields: `id` (unique identifier), `query` (English question), and `cypher` (ground-truth Cypher). From these 346 high-quality samples, we augment the dataset by paraphrasing the English prompts while keeping the Cypher ground truth unchanged. Concretely, for each original English question, we generate 7 diverse paraphrases using Gemini Flash 2.5 under a constrained prompt that enforces semantic preservation (same intent and entities) and encourages variation in phrasing and tone - yielding 2,768 training pairs.

Model and Fine-Tuning

We select **Mixtral-8×7B Instruct** as the backbone LLM. This choice is motivated by (Kouremetis et al. 2025) for AD cypher queries. We fine-tune Mixtral using **QLoRA**, a parameter-efficient method that combines (i) low-rank adapters (LoRA) injected into transformer layers and (ii) quantization of the base model weights to **4-bit** quantization.

Evaluation

We evaluate whether domain-specific fine-tuning improves the reliability of our *tool-grounded* NL→Cypher pipeline for AD defensive analytics: (i) can the model consistently produce Cypher that is *executable* in Neo4j (syntax), and (ii) when executed, does it retrieve results consistent with the intended investigative question (semantics). We benchmark our results against the baseline in (Kouremetis et al. 2025). We restrict the comparison to a single strong open-weight baseline under the same prompting and database conditions in order to isolate the effect of domain-specific fine-tuning. Broader baseline comparisons and component ablations remain important future work.

Experimental Setup

Graphs and databases. We use DBCreator to create AD graphs. We limit the total node count of our AD graphs to 100 in our experiments. We compare the **baseline** open-weight model (Mixtral-8×7B Instruct) as deployed in (Kouremetis et al. 2025) against our **fine-tuned** variant adapted with QLoRA. Both models use the same schema-constrained prompt format and are evaluated on identical databases and query suites. We use the BloodHound-equivalency queries provided in (Kouremetis et al. 2025). These queries represent common AD investigation tasks relevant to defensive triage, including identifying privileged principals (e.g., domain admins), detecting risky configurations, and enumerating credential-theft targets (e.g., kerberoasting).

To account for variability in synthesized AD graphs and the stochasticity of LLM decoding, we generate **five** independent AD databases (`DB1`–`DB5`) using the same synthesis procedure and import each into Neo4j. On each database, we run the entire 10-query suite **ten** times per model and report **per-database averages** results.

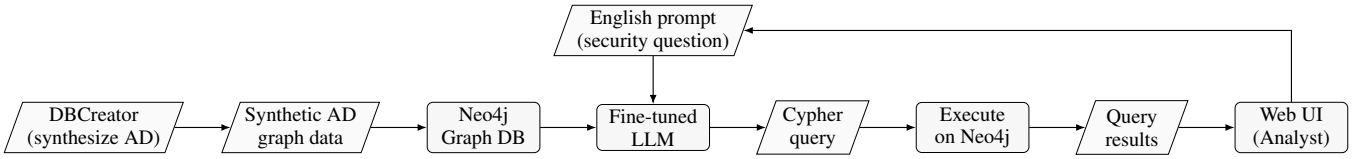

Figure 1: System overview. DBCreator synthesizes an AD graph, which is loaded into Neo4j. An analyst submits a natural-language question via the web UI; the fine-tuned LLM generates a Cypher query constrained by the schema; the query executes on Neo4j and results are returned to the UI.

Metrics

We use two complementary metrics to separate syntactic validity from execution-level correctness on Neo4j/Cypher.

Parse Success Rate (syntax): A generated Cypher query is counted as *parse-successful* if it can be parsed by the Neo4j driver (i.e., it satisfies Cypher grammar and syntax constraints enforced by Neo4j tooling) (Neo4j 2024; Francis 2018). Parse Success Rate measures the fraction of generated queries that parse successfully.

Correct Answer Rate (semantic): To evaluate semantic correctness, we execute each model-generated query on the target AD graph in Neo4j and compare the returned objects to those produced by the ground-truth query (Neo4j 2024; Francis 2018). Following (Kouremetis et al. 2025), a query is considered correct if it returns at least half of the objects in the ground-truth result set. This threshold is useful for comparability with prior work, but in security workflows it may overestimate practical utility because partially correct results can still omit high-risk objects.

Results: Baseline vs. Fine-Tuned

Table 1 reports averaged performance across ten runs for each of five databases. Overall, fine-tuning yields a **large** improvement in syntactic reliability and a **moderate** improvement in execution correctness.

Syntax reliability improves substantially. The baseline Mixtral model achieves Parse Success Rates of approximately 0.78–0.83 across databases, while the fine-tuned model increases these to approximately 0.93–0.95. For defensive workflows, this corresponds to a marked reduction in "tool failure" events where the assistant produces non-executable Cypher, reducing analyst time on debugging query syntax.

Execution correctness improves modestly. Correct Answer Rates increase from approximately 0.29–0.38 (baseline) to approximately 0.38–0.45 (fine-tuned). While this improvement is meaningful, semantic correctness remains the dominant reliability bottleneck: many generated queries are executable yet operationally misleading because they omit critical objects, apply incorrect AD semantics, or mishandle value constraints.

Error Analysis and Human-Aware Design Insights

We observed three recurring failure modes that are particularly important for safe deployment in defensive workflows, where users (or downstream automation) may over-trust

DB	Parse (Base)	Parse (FT)	Correct (Base)	Correct (FT)
DB1	0.78	0.93	0.38	0.44
DB2	0.83	0.94	0.35	0.45
DB3	0.78	0.93	0.36	0.42
DB4	0.80	0.94	0.30	0.39
DB5	0.82	0.95	0.29	0.38
Avg.	**0.80**	**0.94**	**0.34**	**0.42**

Table 1: Condensed baseline vs. fine-tuned performance (averaged over 10 runs per DB).

model outputs. Importantly, these risks can persist even when a query is scored as "correct" under a permissive overlap threshold, reinforcing the need for execution-grounded and domain-specific guardrails.

Domain Admin identification (RID 512 and nesting): For queries that ask for "domain admins" or highly privileged principals, the baseline model sometimes generates patterns that search for usernames or group names containing strings such as `''Domain Admin"` instead of following BloodHound-style membership semantics (e.g., the well-known RID 512 group and nested group relationships). A human analyst reading the natural-language intent and the returned result set may infer that "all domain admins have been enumerated", when in reality critical privileged accounts can be silently omitted. From a risk perspective, this underestimates exposure and can lead to incorrect triage and incomplete remediation plans. Agent systems therefore require rule-based validation of privileged groups and well-known SIDs, as well as domain-specific result checks that confirm domain admin privileges are inferred via group membership and SID semantics rather than naive string matching.

(2) "Unsupported OS" semantic mismatch: For questions about "unsupported operating systems", the model occasionally treats "unsupported" as a literal string match on an `operatingsystem` property instead of enumerating concrete legacy Windows versions that are out of support. An analyst might interpret a small or empty result set as evidence that the environment is free of unsupported hosts, when in fact older versions (e.g., Windows Server releases past end of life) remain deployed but are not matched. This creates a false sense of security and can delay necessary hardening efforts. To mitigate this, human-aware agent designs should couple NL-to-Cypher generation with version lists and apply sanity checks that compare the set of matched versions

against a catalogue of unsupported OS labels.

(3) Temporal constraints and sentinel values: Queries that involve temporal conditions (e.g., "last 90 days") and sentinel values (e.g., timestamps of `0` or `-1` indicating "never") expose another subtle failure mode. The model often produces time-range filters that appear well-formed but omit the additional predicate needed to exclude sentinel values from the result set. To a human, the resulting lists of accounts or sessions look reasonable in size and content, but they can be polluted by entries that should have been excluded or can miss long-lived risky accounts that require special handling. This yields noisy or misleading account lists and increases cognitive load in triage. Schema-aware, value-range checks and explicit tests for sentinel encodings should therefore be built into the agent's post-processing layer, with the UI surfacing when such guardrails have been applied.

Conclusion and Discussion

We presented an LLM-assisted pipeline that translates natural-language security questions into executable Cypher for querying BloodHound-style Active Directory graphs in Neo4j. We curated 346 executable English-Cypher pairs, expanded them to 2,768 paraphrased samples, and fine-tuned an open-weight Mixtral-8×7B model with QLoRA under explicit schema constraints. Our fine-tuned model improved both Parse Success Rate and Correct Answer Rate - indicating that domain tuning can resolve syntax, leaving semantic correctness as the main barrier to safe use in security workflows. Our evaluation is intentionally narrow and should be seen as a preliminary study: it uses synthetic DBCreator graphs capped at 100 nodes and a 10-task BloodHound-equivalency query suite. These choices enable controlled comparison and reproducibility, but they do not establish robustness on larger or noisier enterprise AD deployments.

For human-aware cyber defense agents, this reliability profile argues for a conservative autonomy envelope: NL-to-Cypher should currently be used for *candidate exploration* rather than fully autonomous mitigation, with generated queries treated as suggestions that must be confirmed via known-safe templates, execution-grounded checks, and human inspection. Because semantic errors remain common, human-aware designs must actively calibrate trust, for example by exposing the generated Cypher, highlighting deviations from vetted templates, and attaching confidence signals tied to schema- and value-level guardrails. More broadly, NL-to-Cypher is only one primitive in a larger perceive–reason–act loop; understanding and constraining its failure modes is a prerequisite for safely composing higher-level behaviors.

References

BloodHoundAD. 2023. BloodHound-Tools: DBCreator. https://github.com/BloodHoundAD/BloodHound-Tools/tree/master/DBCreator. Accessed: 2025-12-22.

Dunagan, J.; Zheng, A. X.; and Simon, D. R. 2009. HeatRay: Combating Identity Snowball Attacks Using Machine Learning, Combinatorial Optimization and Attack Graphs. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*. ACM.

Francis, N. e. a. 2018. Cypher: An Evolving Query Language for Property Graphs. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18)*, 1433–1445. New York, NY, USA.

Goel, D.; Ward-Graham, M. H.; Neumann, A.; Neumann, F.; Nguyen, H.; and Guo, M. 2022. Defending active directory by combining neural network based dynamic program and evolutionary diversity optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, 1191–1199. New York, NY, USA: ACM.

Guo, M.; Ward, M.; Neumann, A.; Neumann, F.; and Nguyen, H. 2023. Scalable Edge Blocking Algorithms for Defending Active Directory Style Attack Graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(5): 5649–5656.

Hristidis, V.; Gravano, L.; and Papakonstantinou, Y. 2003. Efficient IR-Style Keyword Search over Relational Databases. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, 850–861. Morgan Kaufmann.

Kouremetis, M.; Dotter, M.; Byrne, A.; Martin, D.; Michalak, E.; Russo, G.; Threet, M.; and Zarrella, G. 2025. OCCULT: Evaluating Large Language Models for Offensive Cyber Operation Capabilities. *arXiv preprint arXiv:2502.15797*.

Li, F.; and Jagadish, H. V. 2014. Constructing an Interactive Natural Language Interface for Relational Databases. *Proceedings of the VLDB Endowment*, 8(1): 73–84.

Munir, S.; and Aldini, A. 2025. Towards Evaluating Large Language Models for Graph Query Generation. In *Computational Science and Computational Intelligence (CSCI 2024)*, CCIS, 30–45. Springer.

Neo4j. 2024. Neo4j Graph Database Platform. https://neo4j.com/. Accessed: 2025-05-20.

Ngo, H. Q.; Guo, M.; and Nguyen, H. 2024. Catch Me if You Can: Effective Honeypot Placement in Dynamic AD Attack Graphs. In *IEEE INFOCOM*. IEEE.

Nickerson, M. 2025a. BloodHound Model Context Protocol Server. https://github.com/mwnickerson/bloodhound_mcp.

Nickerson, M. 2025b. Chatting with Your Attack Paths: An MCP for BloodHound. https://specterops.io/blog/2025/06/04/chatting-with-your-attack-paths-an-mcp-for-bloodhound/. SpecterOps Blog.

Ozsoy, M. G.; Messallem, L.; Besga, J.; and Minneci, G. 2025. Text2Cypher: Bridging Natural Language and Graph Databases. In *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*. Accessed: 2025-12-22.

SpecterOps. 2025. BloodHound Community Edition (Project Page). https://specterops.github.io/bloodhound/. Accessed: 2025-12-22.

Tiwari, A.; Malay, S. K. R.; Yadav, V.; Hashemi, M.; and Madhusudhan, S. T. 2025. Auto-Cypher: Improving LLMs on Cypher Generation via LLM-Supervised Generation-Verification Framework. In *NAACL-HLT (Short Papers)*.

Zhang, Y.; Ward, M.; Guo, M.; and Nguyen, H. 2023. A Scalable Double Oracle Algorithm for Hardening Large Active Directory Systems. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '23, 993–1003. New York, NY, USA: ACM.