Augmenting the Unified Cybersecurity Knowledge Graph Using RAG for Enhanced Threat Detection

Diego Sanchez[1,*,†], Erdogan Dogdu[1] and Roya Choupani[1]

[1]Angelo State University, Department of Computer Science, San Angelo, Texas, USA

Abstract

Large language models (LLMs) often struggle to answer specialized cybersecurity questions correctly due to limited access to structured domain knowledge. This investigation evaluates a retrieval-augmented generation (RAG) approach using the Unified Cybersecurity Knowledge Graph (UCKG) to improve question answering in the cybersecurity domain. The proposed method integrates structured graph queries via a Neo4j database and semantic vector retrieval via ChromaDB with MiniLM embeddings in order to retrieve relevant facts and context for the language model. We evaluated this hybrid system on a gold standard set of 30 cybersecurity questions using four different large language models (Phi-2, Mistral-7B, DeepSeek-Coder, and LLaMA-2 7B). The results show that augmenting these models with the UCKG context improves the answer cosine similarity and detail across all cases. However, the current UCKG lacks support for important frameworks such as CAPEC and MITRE ATT&CK, which limits coverage of certain attack patterns. To address this, future work will incorporate these frameworks and enable multi-hop reasoning to handle more complex queries. Overall, this study demonstrates the effectiveness of combining cybersecurity knowledge graphs with semantic retrieval to improve the performance of language models in domain-specific question answering.

Keywords

Cybersecurity Knowledge Graphs, Graph Augmented Generation, Large language models, LLM

1. Introduction

Cybersecurity knowledge stores are fragmented across multiple data sources and frameworks, making it challenging for automated systems to ingest, synthesize, and retrieve relevant information on demand. Prior efforts to consolidate this knowledge, such as the Unified Cybersecurity Knowledge Graph (UCKG) proposed by Boyer et al [1], demonstrate the value of integrating heterogeneous cybersecurity datasets under a common ontology. The UCKG approach builds on the Unified Cybersecurity Ontology (UCO) to link vulnerabilities, weaknesses, attack vectors, and defensive techniques, paving the way for intelligent security solutions.

However, the task remains in leveraging such a knowledge graph for natural language processing and answering. Large Language Models (LLMs) excel at understanding and generating text, but they often hallucinate or lack up-to-date cyber threat intelligence [2]. Retrieval-Augmented Generation (RAG) has emerged as a strategy to mitigate these issues by equipping LLMs with a grounding knowledge source. In naive implementations, RAGs typically utilize unstructured text corpora, overlooking the rich semantic relations encoded in knowledge graphs. This investigation evaluates the effectiveness of a RAG pipeline that dynamically retrieves context from a Unified Cybersecurity Knowledge Graph to assist LLMs in answering security questions.

We build upon Boyer et al [1]'s UCKG foundation by integrating multiple cybersecurity ontologies (MITRE ATT&CK, D3FEND, CVE, CWE, CAPEC) into a single Neo4j graph and using it as a source of truth for question answering. The goal is to determine whether RAG grounded in a unified graph can improve answer accuracy and reliability for cybersecurity queries, compared to LLMs used in isolation.

We review the related work in Section 2. Graph RAG implementation details are presented in Section 3. We evaluate and discuss the performance of the system in Section 4 and conclude in Section 5.

1st Workshop on Graphs Across AI: From Structural Reasoning to Foundation Models, held in conjunction with the IEEE World Congress on Computational Intelligence (WCCI 2026), June 21–26, 2026, Maastricht, The Netherlands

dsanchez35@angelo.edu (D. Sanchez); erdogan.dogdu@angelo.edu (E. Dogdu); roya.choupani@angelo.edu (R. Choupani)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-08

CEUR-WS.org/Vol-4319/paper5.pdf

2. BACKGROUND

2.1. Knowledge Graphs in Cybersecurity

Cybersecurity knowledge graphs (CKGs) provide a structured method for capturing relationships between threat indicators, vulnerabilities, and mitigation strategies. CKGs have become a popular approach to organize and utilize data in cybersecurity. Sikos [3] outlines the core benefits of CKGs, including their ability to unify fragmented threat intelligence across diverse sources. Wu et al [4] expands on this by incorporating large language models (LLMs) to refine the credibility and structure of graph nodes and edges, improving trust in automated CTI pipelines. However, as Fieblinger et al [5] notes, the scalability of these systems is limited by the difficulty of extracting meaningful insights from large volumes of unstructured data. Early cybersecurity KGs often focused on specific domains or standards. For example, Iannacone et al [6] developed STUCCO, an ontology-driven graph consolidating structured threat intelligence sources such as CVEs and attack reports. Syed et al [7] introduced the Unified Cybersecurity Ontology (UCO) to align key schemas, those being: CVE (Common Vulnerabilities and Exposures), CWE (Common Weakness Enumeration), CAPEC (Common Attack Pattern Enumeration and Classification), and STIX – into a single ontology.

Numerous other ontologies and knowledge bases have since been proposed, each covering aspects like malware attributes, intrusion kill chains, and defensive measures. With this proliferation in cybersecurity ontologies rises a persistent challenge in interoperability and unification of these knowledge sources. Boyer et al. [1] addressed this challenge by integrating heterogeneous cybersecurity data into a Unified Cybersecurity Knowledge Graph using UCO, enabling advanced analytics over a consolidated CKG. However, querying and reasoning over such graphs remains difficult for the layperson. Natural language interfaces powered by LLMs offer a promising solution that enables intuitive access to KG-based knowledge.

2.2. CTI Integration and Graphs LLMs

Recent approaches have demonstrated the effectiveness of combining Cyber Threat Intelligence (CTI) with knowledge graphs through Retrieval-Augmented Generation (RAG). Zhang et al [8] developed an architecture that uses instruction prompting and in-context learning to convert CTI into usable graph elements, improving both recall and threat coverage. Similarly, Kurniawan et al [9] proposed CyKG-RAG, a hybrid system that leverages retrieval-augmented reasoning to link external evidence to dynamic graph nodes. These efforts underscore the promise of AI-enhanced automation in maintaining up-to-date threat intelligence graphs. Cyber Threat Intelligence (CTI) encompasses frameworks like MITRE ATT&CK (for adversary patterns) and D3FEND (for defensive countermeasures), as well as standards for indicator exchange (STIX/TAXII). These structured resources enable reasoning about threats, but traditionally require expert users or specialized query languages - CYPHER being a prominent one. Integrating CTI knowledge into LLM reasoning is an emerging area. Some studies have looked at using LLMs to generate graph queries or to traverse knowledge graphs via natural language instructions [10].

Large language models (LLMs) are increasingly used to augment knowledge graph systems by performing tasks such as node expansion, relationship validation, and threat scenario prediction. Wu et al [4] introduced KGV, a framework for leveraging LLMs to assess the credibility of threat intelligence and perform fine-grained relationship inference. [5] explored few-shot and prompt engineering techniques to lower the cost of adapting LLMs to cybersecurity-specific domains. These studies demonstrate the feasibility of using LLMs not only as text generators but as intelligent filters and validators for structured threat data.

Others have examined using graph embeddings and neural symbolic reasoning to guide LLMs [11]. A key insight is that LLMs can benefit from graph context – an LLM knowing that a certain CVE maps to a CWE category or is linked to a MITRE ATT&CK technique can help the model provide a more comprehensive answer. Graph reasoning with LLMs can take forms such as: having the LLM generate intermediate structured representations (like triples or paths) as part of chain-of-thought, or using graph algorithms to fetch multi-hop neighbors of an entity mentioned in the query. While our current

work does not implement full multi-hop reasoning, it aligns with this direction by retrieving nodes from a richly interlinked KG. We anticipate that as LLMs become better at tool use and multi-step reasoning, they could interact with cybersecurity KGs more directly by iteratively querying the graph for missing pieces of information. This investigation contributes to the groundwork by demonstrating that even a straightforward RAG pipeline in tandem with the UCKG can measurably improve answer accuracy on cybersecurity QA, highlighting the potential of deeper graph-based reasoning.

2.3. Retrieval-Augmented Generation (RAG)

RAG is a technique where an LLM is augmented with an external retriever to fetch relevant context (e.g. documents, facts) which the model can ground itself on when generating answers. [12] formalized this approach for knowledge-intensive NLP tasks, demonstrating that even large models benefit from conditioning on retrieved evidence. By injecting pertinent information at query time, RAG can reduce hallucinations and provide up-to-date knowledge without exhaustive retraining.

In cybersecurity, RAG has been applied to tasks like threat report summarization and indicator extraction, but typically using unstructured text sources such as documentation or incident databases. Recent research is exploring graph-based RAG variants ("GraphRAG"), which retrieve not just documents but nodes or subgraphs from a knowledge graph to enrich the context. Kurniawan [9] proposed CyKG-RAG, a framework that integrates a cybersecurity knowledge graph with RAG for threat analysis. By leveraging the relationships between entities, their system provides contextually relevant facts to the LLM and improved detection of attack patterns. Such graph-enhanced RAG approaches have shown promise in domains like biomedicine and recommendation systems as well [11][13], but are only beginning to be applied to cybersecurity. Our work builds on these ideas by using a unified graph of cybersecurity knowledge as the retrieval source, aiming to improve the quality of answers that LLMs produce for security questions.

3. UCKG-RAG IMPLEMENTATION

We constructed the Unified Cybersecurity Knowledge Graph (UCKG)[1] in Neo4j to serve as the knowledge base. The graph incorporates nodes and relationships from several authoritative cybersecurity data sources: CVE[2] (software vulnerabilities), CWE[3] (classes of software weaknesses), CAPEC[4] (attack patterns), MITRE ATT&CK[5] (adversary tactics/techniques), and MITRE D3FEND[6] (defensive techniques). Each node in the graph represents an entity such as a specific vulnerability (e.g., CVE-2021-44228), a weakness category (e.g., CWE-79), an attack pattern (e.g., CAPEC-112), etc. Properties on these nodes include textual descriptions (i.e ucocweSummary denoting a summary of a particular weakness in a CWE), references, and identifiers. Edges between nodes encode known relationships - a CWE, for instance, specifies a "has weakness" relationship than links a CVE(s) to a CWE that characterizes their root cause. A visualization of a portion of a CVE/CWE node with its 5 nearest neighbors can be seen in following Figure 1 and Figure 2:

By fusing the aforementioned sources, the knowledge graph provides a unified view – a vulnerability node in our graph is not only described by its CVE entry but also connected to related weaknesses. This unified schema (aligned loosely with UCO) ensures that information useful for answering security questions can be traversed within a single graph. All graph data was loaded into Neo4j 4.4.0 Enterprise Edition[7], enabling fast graph queries.

[1]UCKG: https://github.com/edogdu/UCKG
[2]CVE: https://www.cve.org
[3]CWE: https://cwe.mitre.org
[4]CAPEC: https://capec.mitre.org
[5]ATT&CK: https://attack.mitre.org
[6]D3FEND: https://d3fend.mitre.org/
[7]Neo4j: https://neo4j.com/

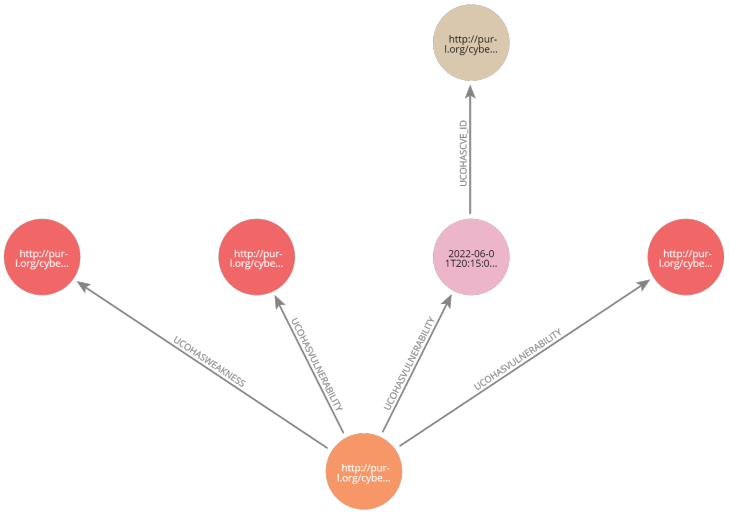


Figure 1: Graph illustrating ontology relationships among Common Vulnerabilities and Exposures (CVE) entries in the Unified Cybersecurity Knowledge Graph (UCKG). Nodes represent CVE entities; edges depict semantic relationships (e.g., hierarchical links, related threats). The figure visualizes how CVE vulnerabilities are interconnected in the UCKG graph-based context.

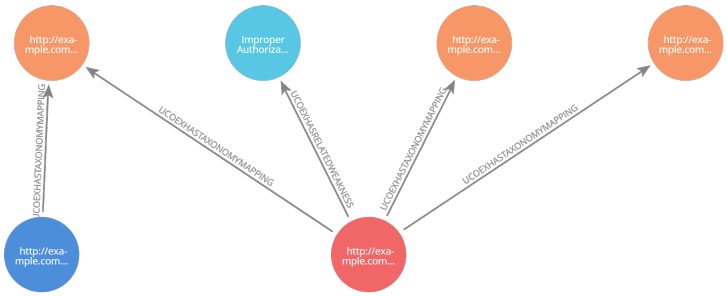


Figure 2: Graph illustrating ontology relationships among Common Weakness Enumeration (CWE) entries in the Unified Cybersecurity Knowledge Graph (UCKG). Nodes represent CWE entities; edges depict semantic relationships (e.g., hierarchical links, related categories). The figure visualizes the graph-based context of CWE vulnerabilities within the UCKG.

3.1. RAG Pipeline Design

The question-answering system uses a Retrieval-Augmented Generation approach, wherein an LLM is coupled with a retriever that fetches relevant knowledge graph content to ground the LLM's answers. We adopted a hybrid retrieval strategy combining semantic search with graph traversal. A high-level illustration of the pipeline can be seen in Figure 3. The pipeline consists of the following steps for each query:

1. **Question Embedding**: The input question (in natural language) is first converted into a dense vector representation using a SentenceTransformer model (specifically all-MiniLM-L6-v2 [8], a lightweight 6-layer transformer optimized for semantic similarity tasks). This embedding captures

[8] MiniLM-L6-v2: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

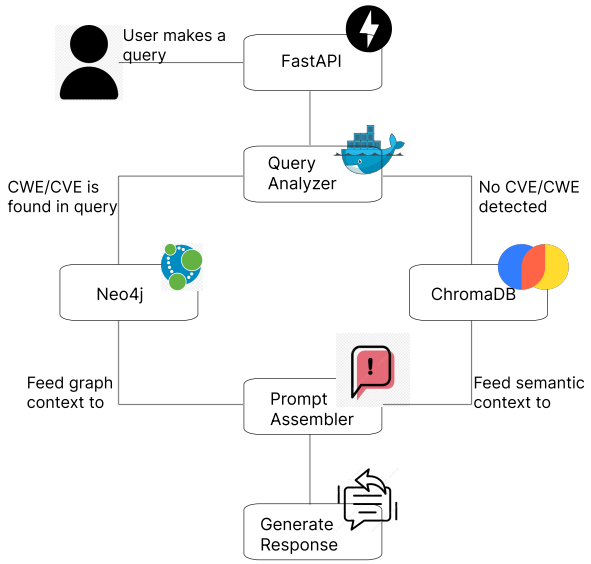


Figure 3: System architecture of the UCKG-RAG pipeline. Queries are routed through a semantic analyzer and assigned to a ChromaDB-based vector search or a Neo4j-based graph search based on presence of structured identifiers (CVE/CWE).

the contextual meaning of the question.

2. **Semantic Search in Vector Store**: We maintain a vector index of knowledge graph content using ChromaDB[9], an open-source vector database. Each node in the cybersecurity KG has been pre-embedded using the same MiniLM model, based on its description and key attributes. Given the question embedding, we perform a similarity search in ChromaDB to retrieve the top-k (e.g., $k = 5$) most relevant nodes from the graph.
   If a question asks about a specific CVE, the semantic search might retrieve that CVE node as well as related CWE nodes or similar vulnerabilities. This step allows us to handle lexical variation – the question need not exactly match node names, as the embedding search can identify conceptually related nodes (e.g., a query about "Log4Shell vulnerability" will retrieve the corresponding CVE node even if not mentioned verbatim).
3. **Graph Context Retrieval**: Optionally, for each node retrieved, we also pull its immediate neighbors in the graph directly from Neo4j (and their descriptions) to enrich the context. This leverages the KG's relationships: for instance, if a CVE node is retrieved, its linked CWE nodes may provide additional insight to answer why the vulnerability is important or how it is exploited. The system assembles a context passage containing the descriptions of the top nodes and key neighbors.
4. **LLM Generation with Context**: The retrieved context is provided to a prompt including the original question to form an augmented prompt for the LLM. We tested four different language models as the core of the pipeline:
   - microsoft/phi-2[10] - a 2.7B-parameter model from Microsoft, known for strong performance despite its small size;
   - meta-llama/Llama-2-7b-chat-hf[11] – a 7B-parameter chat-tuned model by Meta, representing a powerful baseline for knowledge tasks;
   - deepseek-ai/deepseek-coder-6.7b-base[12] – a 6.7B model oriented towards code and technical Q&A, included to assess a domain-specialized model;
   - mistralai/Mistral-7B-v0.1[13] – an open 7B model with strong general performance.

[9]ChromaDB: https://www.trychroma.com
[10]Microsoft Phi2: https://huggingface.co/microsoft/phi-2
[11]Llama2-7b-chat-hf: https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
[12]DeepSeek Coder 6.7b-base: https://huggingface.co/deepseek-ai/deepseek-coder-6.7b-base
[13]Mistral 7B v0.1:https://huggingface.co/mistralai/Mistral-7B-v0.1

Each model was prompted with the same formatted input: a system message explaining its role, the injected context from the KG (as reference material), and the user's question. The model then generates an answer, using the provided context to increase accuracy.

5. **Answer Output**: The model's answer is captured and returned as the final output. We run two modes: RAG-augmented mode, where context is provided as above, and a No-RAG baseline, where the model is given only the question (to observe how it performs without any knowledge injection).

By combining the unified KG with semantic retrieval, our pipeline aims to supply the LLMs with structured cybersecurity-specific knowledge in real-time. Neo4j's role is to store and organize the data, while ChromaDB enables fast similarity search over the graph's content. This design ensures that even relatively small models can leverage a broad base of security knowledge far beyond their internal training data. It effectively "injects" facts about specific CVEs, mitigations, and so on, into the model's context window just when needed, rather than relying on the model to have memorized those facts. The idea behind this implementation is that this will yield more precise answers, especially for detailed or less-common security questions, compared to the model alone.

3.2. Evaluation Setup

To evaluate the effectiveness of the RAG approach, we curated a gold-standard set of question-answer pairs covering a range of cybersecurity topics. In total, 30 questions were assembled, each categorized by type: CVE-specific (10 questions), CWE-specific (10 questions), or general security (10 questions). The CVE questions ask about particular software vulnerabilities (e.g., "What is the vulnerability described in CVE-2021-44228?"), expecting answers with the vulnerability's nature and impact.

The CWE questions target definitions of weakness classes or their implications (e.g., "How does CWE-79 relate to cross-site scripting attacks?"). The general questions are broader, often bridging concepts (e.g., asking for vulnerabilities in a product or common issues in a technology, which requires knowledge of multiple CVEs/CWEs).

Tables 1, 2, and 3 present the full set of evaluation questions used in this study, categorized by CVE, CWE, and general cybersecurity topics, respectively. Each of these is paired with its corresponding gold-standard answer—derived from authoritative sources such as NVD vulnerability descriptions and CWE database entries—which are provided in Tables 4, 5, and 6. These reference answers represent the ideal responses that a correct system output should contain. Using this dataset, each of the four large language models (LLMs) was evaluated across all 30 questions under two conditions: one with retrieval-augmented generation (RAG) providing knowledge graph context, and one without. This setup yielded a total of eight evaluation runs (four models, each tested in two modes).

Table 1
CVE-Based Evaluation Questions

ID	Question
1	What is the vulnerability described in CVE-2021-44228?
2	Which component is affected by CVE-2017-0144?
3	What is the impact of CVE-2022-22965 in Spring applications?
4	What type of vulnerability is CVE-2018-0171 in Cisco IOS?
5	How is CVE-2023-23397 exploited in Microsoft Outlook?
6	Which CWE is CVE-2020-0601 mapped to?
7	Is CVE-2021-34527 remotely exploitable?
8	What mitigation was recommended for CVE-2022-30190?
9	What attack technique does CVE-2020-0796 enable?
10	Was CVE-2014-0160 considered a confidentiality flaw?

To quantify performance, we adopted cosine similarity between the model's answer and the gold-standard answer as the evaluation metric. Cosine similarity is computed in the embedding space of the

Table 2
CWE-Based Evaluation Questions

ID	Question
11	What does CWE-287 refer to?
12	How does CWE-79 relate to cross-site scripting attacks?
13	Which weakness is CWE-89 and what is its impact?
14	What class of issues does CWE-20 cover?
15	What is the core issue described by CWE-22?
16	How is CWE-352 exploited in web applications?
17	What makes CWE-200 a serious vulnerability?
18	What protections does CWE-310 aim to enforce?
19	What is the risk associated with CWE-502?
20	How does CWE-798 undermine authentication?

Table 3
General Evaluation Questions

ID	Question
21	What are the security flaws in OpenBSD's IP stack?
22	How do attackers exploit outdated SSL libraries?
23	What are the consequences of missing input validation?
24	What types of flaws are common in IoT devices?
25	Which weaknesses make cross-site request forgery possible?
26	What are the main causes of buffer overflows?
27	What vulnerabilities have been found in SMB protocol implementations?
28	What types of injection attacks affect databases?
29	How are access control flaws exploited in practice?
30	What are common web application weaknesses in CWE?

same SentenceTransformer (all-MiniLM-L6-v2) and yields a continuous score from −1 to 1 indicating how similar the semantic content of the answers is to the reference. A score of 1.0 would mean the model response is essentially identical in meaning to the gold answer, while a score near 0 means the response is off-topic or entirely incorrect. This embedding-based evaluation is more forgiving to phrasing differences than exact match, and it aligns with judging semantic correctness. We calculated the cosine similarity for each question's Q&A pair. Then, for each model, we computed the average similarity across all 30 questions in RAG vs. no-RAG mode.

While an exact match or human expert evaluation would be ideal, the cosine similarity provides an automated, quantitative proxy to compare performance. We also inspected the raw outputs for qualitative analysis, noting where models made factual errors or omissions. The evaluation specifically probes whether providing graph-derived context improves the models' ability to answer correctly. We expect the benefit to be most pronounced on questions requiring specific factual knowledge (e.g., details of a CVE), where an unsupported LLM might hallucinate or give vague answers, whereas a supported one can draw from the context.

By covering both narrow (CVE, CWE) and broad questions, we also gauge the limits of the approach – for example, whether the models can synthesize multiple retrieved facts to answer integrative questions.

4. RESULTS

The retrieval-augmented approach improved answer quality for all four LLMs, as evidenced by higher cosine similarity scores in the with-RAG condition after retrieving the 3 nearest neighbors from the source node. Figure 4 summarizes the average cosine similarity achieved by each model with and without knowledge graph augmentation.

Table 4
CVE-Based Gold Standard Answers

ID	Answer
1	CVE-2021-44228, also known as Log4Shell, is a remote code execution vulnerability in the Apache Log4j 2 library. It allows unauthenticated attackers to execute arbitrary code by manipulating log messages.
2	CVE-2017-0144 affects Microsoft's SMBv1 server component and is exploited by the EternalBlue exploit. It enables remote code execution via specially crafted packets.
3	CVE-2022-22965, or Spring4Shell, allows remote attackers to execute arbitrary code in Spring MVC applications running on Java 9+ under certain conditions. It exploits data binding features in Spring's RequestMappingHandlerAdapter.
4	CVE-2018-0171 is a command injection vulnerability in Cisco IOS XE software due to improper validation of arguments in Smart Install. It allows an unauthenticated attacker to execute arbitrary code on affected devices.
5	CVE-2023-23397 is exploited through a specially crafted email that triggers code execution when retrieved and processed by Microsoft Outlook. No user interaction is required, making it a zero-click vulnerability.
6	CVE-2020-0601 maps to CWE-295: Improper Certificate Validation. It impacts Windows CryptoAPI and allows spoofing of ECC certificates, enabling man-in-the-middle attacks.
7	Yes, CVE-2021-34527 (PrintNightmare) is remotely exploitable. It affects the Windows Print Spooler service and can allow remote code execution with SYSTEM privileges.
8	For CVE-2022-30190 (Follina), Microsoft recommended disabling the MSDT URL protocol as a workaround. This prevents invocation of the vulnerable Microsoft Support Diagnostic Tool.
9	CVE-2020-0796, known as SMBGhost, enables remote code execution by exploiting a buffer overflow in Microsoft's SMBv3 protocol. It can be used for lateral movement within networks.
10	Yes, CVE-2014-0160 (Heartbleed) allowed attackers to read sensitive information from memory, such as private keys and session tokens. This was a severe confidentiality issue in OpenSSL.

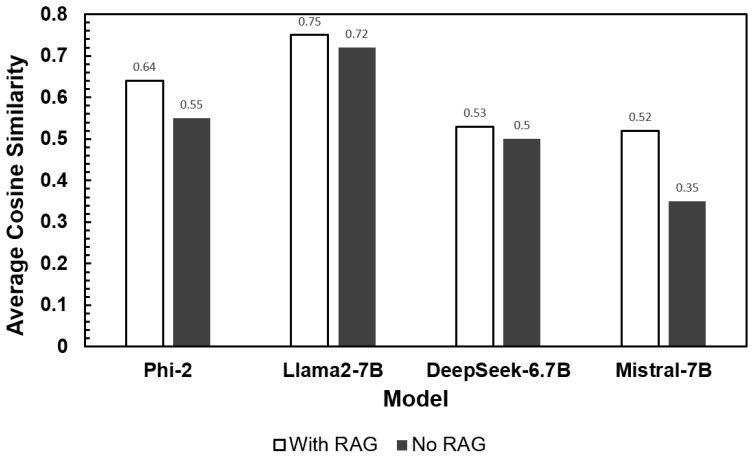


Figure 4: Average cosine similarity for each model when returning context from 3 nearest neighbors, with and without RAG context. Results were obtained from by feeding each model 30 questions of varying type. This demonstrates an increase in model answer similarity to the "golden answer."

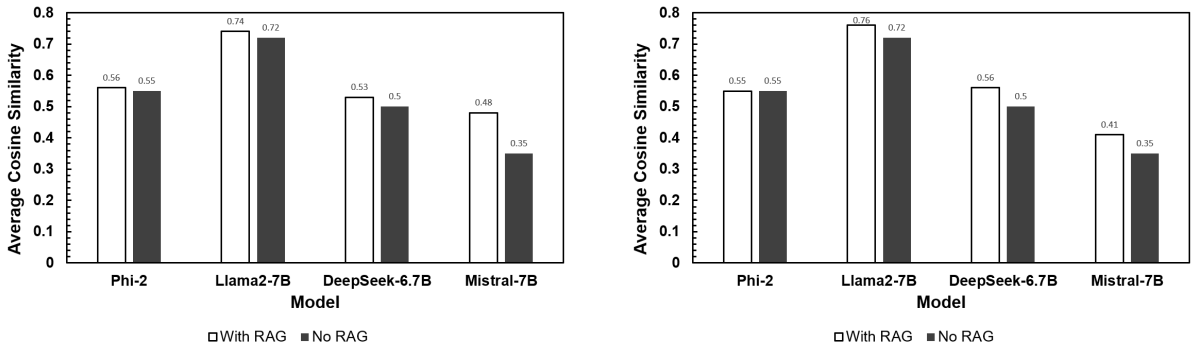
Figure 4 illustrates these results, comparing the average similarity scores with vs. without RAG for each model. We observe that LLaMA-2 7B achieved the highest scores overall (above 0.75 with RAG), indicating it often produced answers close to the reference. Phi-2's score around 0.64 with RAG is notable given its much smaller size; it outperformed the larger DeepSeek and Mistral models in our domain-specific test, suggesting that certain smaller models fine-tuned for general knowledge can be quite effective when augmented with the right context. Mistral 7B's no-RAG score (0.35) was the lowest of all runs, reflecting that without context it frequently failed to recall or infer the correct information. With retrieval, its score rose to 0.52, showing that context narrowed the gap, though it still lagged

Table 5
CWE-Based Gold Standard Answers

ID	Answer
11	CWE-287 refers to Improper Authentication, where the system fails to properly verify the identity of users or devices. This can lead to unauthorized access to protected resources.
12	CWE-79 is the category for Cross-site Scripting (XSS), where untrusted input is injected into a web page without proper validation or encoding. It allows attackers to execute scripts in users' browsers.
13	CWE-89 refers to SQL Injection, which occurs when user input is improperly sanitized before being included in SQL queries. This allows attackers to execute arbitrary SQL code against a database.
14	CWE-20 covers Improper Input Validation, where a program does not adequately validate input from users or external sources. This can lead to injection, buffer overflows, or unexpected behavior.
15	CWE-22, Path Traversal, involves improper restriction of file path inputs, allowing attackers to access files outside the intended directory. It often arises when user input is used to construct file paths.
16	CWE-352 describes Cross-Site Request Forgery (CSRF), where attackers trick users into submitting unauthorized requests using their authenticated session. It exploits the trust a web application has in the user's browser.
17	CWE-200 involves Information Exposure, where sensitive data is disclosed to unauthorized users. This can lead to breaches of confidentiality, such as exposing credentials or internal system details.
18	CWE-285 is Improper Authorization, where access control is not correctly enforced, while CWE-287 is Improper Authentication, where the identity of users is not verified. The former governs what users can do, the latter governs who they are.
19	CWE-362 covers race conditions like Time-of-Check to Time-of-Use (TOCTOU), where the state of a resource changes between check and use. This can allow attackers to manipulate file or memory state unexpectedly.
20	CWE-416 refers to Use After Free, where a program continues to use memory after it has been released. This can lead to crashes, data corruption, or arbitrary code execution.

behind the others. This trend supports the idea that the less pre-trained domain knowledge a model has, the more it stands to gain from RAG.

The effect of increasing context provided to the model was then tested by increasing the amount of nearest neighbors retrieved. We measured the average cosine similarity between each model's answer and a golden reference across 30 cybersecurity questions for each run in the same manner as shown in Figure 5a. Using the top 5 nearest neighbors improved similarity scores for all models, with LLaMA2-7B achieving the highest performance overall. Notably, Mistral-7B showed substantial gains when RAG was applied. However, the cosine similarity averages decreased when comparing the results to Figure 4. These results indicate that structured context retrieval enhances model alignment with reference answers but show that supplying excess context interferes with performance.



(a) 5 nearest neighbors.

(b) 10 nearest neighbors.

Figure 5: Average cosine similarity for each model when returning context from 5 or 10 nearest neighbors, with and without RAG context. Results were obtained from by feeding each model 30 questions of varying type. This demonstrates a slight increase in model answer similarity to the "golden answer."

Table 6
General Gold Standard Answers

ID	Answer
21	OpenBSD has been affected by vulnerabilities such as format string issues in su and ssh, and denial of service via kernel panic through ipintr() and ipq. These map to weaknesses like CWE-134 and CWE-400.
22	Microsoft Outlook has been targeted by CVEs such as CVE-2023-23397, which allowed zero-click code execution via calendar invites. These vulnerabilities often involve CWE-287 and CWE-77.
23	Apache Log4j has been notably impacted by CVE-2021-44228, which allows remote code execution via JNDI injection. This vulnerability maps to CWE-502 and CWE-20.
24	SMB protocol implementations have been exploited through CVEs like CVE-2017-0144 and CVE-2020-0796. These are linked to weaknesses such as buffer overflows (CWE-119) and improper input validation (CWE-20).
25	Improper input validation can lead to a range of weaknesses including buffer overflows, injection attacks, and denial of service. Common CWEs involved are CWE-20, CWE-119, and CWE-89.
26	Spring Framework vulnerabilities include CVE-2022-22965 (Spring4Shell) and CVE-2018-1270. These relate to improper input handling and insecure deserialization, such as CWE-20 and CWE-502.
27	Authentication bypass attacks commonly exploit CWE-287: Improper Authentication and CWE-306: Missing Authentication for Critical Function. They may also involve manipulation of session state or credentials.
28	The Windows Print Spooler service has been exploited via vulnerabilities like CVE-2021-34527 (PrintNightmare). These often map to CWE-269: Improper Privilege Management and CWE-269.
29	Information disclosure vulnerabilities are often categorized under CWE-200. Real-world examples include Heartbleed (CVE-2014-0160) and misconfigured cloud storage buckets.
30	Web applications are frequently affected by XSS (CWE-79), CSRF (CWE-352), SQL injection (CWE-89), and improper authentication (CWE-287). These issues arise due to poor input handling and lack of secure session management.

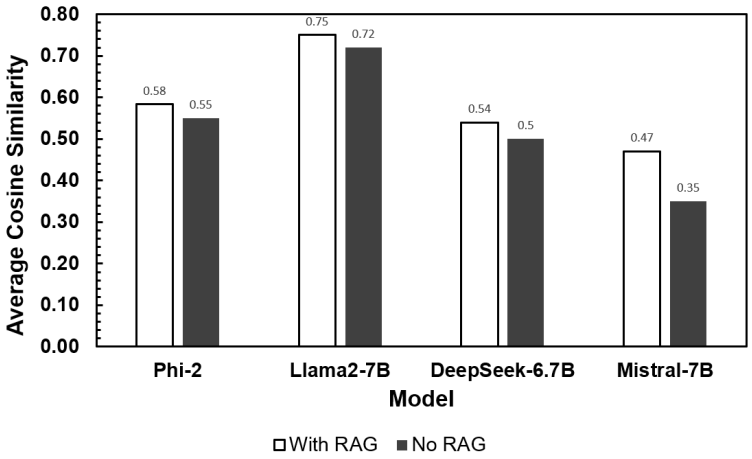


Figure 6: Average cosine similarity for each model with and without RAG context across all runs. Results were obtained from by feeding each model 30 questions of varying type. This demonstrates an increase in model answer similarity to the "golden answer" when retrieving nearest neighbor context.

Figure 5b illustrates the effect of expanding context retrieval to the 10 nearest neighbors. Compared to the 5-nearest setup, average cosine similarity scores declined for most models, particularly lower-parameter ones like Phi-2 and Mistral-7B. While LLaMA2-7B remained the top performer, the marginal gains observed with RAG in the previous configuration diminished. This suggests that retrieving more documents introduces additional noise or less relevant context, diluting the model's ability to focus on high-signal content. The trade-off between context breadth and relevance becomes more apparent as the number of neighbors increases.

Figure 6 shows that when averaging results across all nearest neighbor retrieval settings, we observe

a consistent increase in cosine similarity for all models when RAG context is applied. Despite the introduction of some noise at higher retrieval depths—particularly with 10-nearest neighbors—the overall trend confirms that augmenting language models with knowledge graph-derived context improves answer quality. This reinforces the value of RAG, even when some retrieved context may be less relevant, as the structured information still helps guide the models closer to semantically correct responses.

4.1. Per-Question Trends

We analyzed the cosine similarity for each individual question to see where RAG helped the most. In general, questions about specific vulnerabilities (the CVE set) showed the clearest improvement. For example, for the question on CVE-2021-44228 ("Log4Shell"), models without RAG assistance sometimes conflated it with unrelated issues or gave partially correct answers. With the KG context (which included the CVE's description and links to CWE-502, etc.), all models were able to correctly identify it as a remote code execution in Log4j. The similarity scores for that question jumped accordingly. On the other hand, some of the general questions ("What security weaknesses are often found in web applications?") saw high scores both with and without RAG for the stronger models – likely because such broad questions overlap with commonly known security knowledge (XSS, CSRF, SQLi are well-known issues). In those cases, LLaMA-2 often answered well even without help, and indeed the marginal gain from RAG was small.

However, the smaller models without RAG sometimes missed key points (for instance, Mistral without RAG failed to list any specifics for the web app weaknesses question, yielding near-zero similarity, whereas with context listing CWE-79, CWE-89 etc., it scored much higher). We also noticed that the CWE definition questions (e.g., "What does CWE-287 refer to?") were answered fairly reliably by all models in both modes – likely because these are definitional and many models have some memory of common CWE names. Still, RAG ensured that the answers were phrased correctly as per the official definitions (including details like "Improper Authentication" wording), improving consistency. In summary, RAG had the most impact on factual, specific queries about CVEs and on any query where the model would otherwise lack detail. For broad or very common knowledge questions, a well-trained model like LLaMA-2 could answer reasonably on its own, but even there RAG sometimes added an extra fact or example, increasing the answer's completeness.

4.2. Error Analysis

Despite overall improvements, the models did not achieve perfect similarity. We manually examined cases with low similarity even after RAG. In some instances, the retrieved context was only tangentially related to the question, which led the model astray. A question about a specific old CVE might retrieve a related but different vulnerability due to embedding similarities, and the model then mixed up details in its answer. This highlights a limitation of pure semantic search – without a precise graph query or filtering by entity type, the retriever can supply irrelevant info that is sometimes logically incorrect. Another observed issue was incomplete answers: a model might recall or use only part of the provided context. A question asking "Which vulnerabilities have affected SMB protocol implementations?" expects two CVEs in the answer. With RAG, the context did list both CVE-2017-0144 (EternalBlue) and CVE-2020-0796 (SMBGhost), but some models only mentioned one of them in the generated answer. This suggests that the prompt format or the model's ability to integrate multiple facts has room for improvement.

In a few cases, the models produced hallucinations even in RAG mode – e.g., DeepSeek in one run cited a fictitious CVE ID that wasn't in the context. Such errors indicate that if the model isn't sufficiently constrained, it can still introduce unsupported content. The overall trend was that hallucinations decreased with RAG. The cosine similarity metric, while useful, may not capture all aspects of answer quality; an answer could have the right facts but poor grammar and incorrect logic, which our method would rate highly. In qualitative terms, the RAG-based answers were more aligned with the style and content of the reference, whereas no-RAG answers had more variability. Notably, LLaMA-2's answers

in RAG mode were often very close to the gold standard phrasing (sometimes paraphrased) – indicating that the context was directly used. Phi-2 occasionally injected context verbatim from the graph data, which can boost similarity but might be a concern for "naturalness" of the model. These observations will inform how we refine the prompting in future iterations.

5. CONCLUSION

This work presented a retrieval-augmented Q&A system enhanced by a Unified Cybersecurity Knowledge Graph, and demonstrated its effectiveness in improving the accuracy of answers to security questions. By unifying data from CVE, CWE, CAPEC, ATT&CK, and D3FEND into a single Neo4j graph, and using semantic vector search and specific querying to retrieve relevant nodes as context, we enabled LLMs to produce more factually correct and detailed answers. The results, evaluated on a diverse set of 30 questions, showed clear benefits of the RAG approach – particularly allowing smaller models like Microsoft's Phi-2 and Mistral-7B to perform better in a specialized domain. The system capitalizes on the strengths of both components: the knowledge graph provides a structured, authoritative information repository, and the LLM provides fluent natural language understanding and generation. Together, they address some of each other's weaknesses: the KG alone is not user-friendly for Q&A, and the LLM alone can be uninformed or hallucinate. This led to answers that are not only more accurate but also contextual. However, there is considerable room to grow. The encouraging results obtained here form a strong foundation upon which future research and development can build more advanced graph-powered, AI-driven cybersecurity intelligence systems.

6. FUTURE WORK

A key future direction is to extend multi-ontology support even further. Our current KG, while unified, could be enriched with additional data sources such as the Common Platform Enumeration (CPE) for platform-specific vulnerability context, or threat actor databases to answer questions about who exploits a given CVE. Another authoritative datasource worth investigating are After-Action Reports (AARs). Incorporating AARs into the UCKG can offer deep insights into real-world incident responses and operational failures. Incorporating vulnerability severity metrics (CVSS scores) and exploit kit information could also enable the system to answer risk assessment questions. Adding a decision tree to rank answers retrieved by the RAG can be beneficial in retrieving relevant context when pulling many nearest neighbors, reducing noise. By broadening the KG, the system can handle a wider array of queries. In the next iteration, we envision the LLM not just receiving a blob of context, but actively guiding graph traversals. For instance, if a question asks, "How is vulnerability X related to attack technique Y?", the system could perform a multi-hop path search in the graph between X and Y, and then have the LLM explain that path in words. Techniques like sequential prompting or using the LLM to generate Cypher queries could allow more complex questions to be answered.

Acknowledgment

This research was supported by the U.S. Department of War, Army Research Laboratory (ARL), under Grant No. W911NF-24-2-0180 and TTU Subaward 21C284-01 for Angelo State University. The authors would like to express their gratitude for the support provided by the DoW and ARL for the project titled "Research and Workforce Development in the Area of Cybersecurity". The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of War or the U.S. Government.

Declaration on Generative AI

ChatGPT (OpenAI) was employed solely for polishing manuscript text, including grammar checks, word correction, and similar stylistic improvements. No part of the scientific content, research methodology, or analytical results was generated by AI.

References

[1] A. Boyer, E. Dogdu, R. Choupani, J. S. Watson, D. Sanchez, A. Ametu, Toward a unified cybersecurity knowledge graph: Leveraging ontologies and open data sources, in: Proc. of the Southwest Data Science Conference 2024, Springer Nature, 2024, pp. 17–33.

[2] R. Zhang, Y. Lyu, Y. Li, Y. Fu, Hallucination in large language models: A survey, arXiv preprint arXiv:2305.10174 (2023).

[3] L. F. Sikos, Cybersecurity knowledge graphs, Knowledge Graphs and Big Data Processing (2023).

[4] Z. Wu, F. Tang, M. Zhao, Y. Li, Kgv: Leveraging large language models for cyber threat intelligence credibility and structure assessment, arXiv preprint arXiv:2401.xxxxx (2024).

[5] R. Fieblinger, M. T. Alam, N. Rastogi, Actionable threat intelligence extraction via prompt-engineered large language models, arXiv preprint arXiv:2402.xxxxx (2024).

[6] M. D. Iannacone, B. Stoner, C. Brown, S. O'Boyle, R. S. Bridges, Developing an ontology for cyber security knowledge graphs, in: Proc. 10th ACM Workshop on Artificial Intelligence and Security (AISec), 2015, pp. 1–10.

[7] Z. Syed, A. Padia, T. Finin, L. Mathews, A. Joshi, Uco: A unified cybersecurity ontology, in: Proc. AAAI Workshop on Artificial Intelligence for Cyber Security, 2016.

[8] Y. Zhang, T. Du, Y. Ma, X. Wang, Y. Xie, X. Shen, Instruction-driven cti structuring: Enhancing threat graph coverage with in-context learning, arXiv preprint arXiv:2401.10293 (2024).

[9] K. Kurniawan, E. Kiesling, A. Ekelhart, Cykg-rag: Towards knowledge-graph enhanced retrieval augmented generation for cybersecurity, in: Proc. RAGE-KG 2024 Workshop at ISWC, volume 3950 of *CEUR Workshop Proc.*, 2024, pp. 51–64.

[10] R. Ye, C. Zhang, R. Wang, S. Xu, Y. Zhang, Language is all a graph needs, arXiv preprint arXiv:2308.07134 (2023).

[11] L. D. Manso, et al., Knowledge graphs meet llms: Comparative analysis of graphrag, llm, and rag in biomedical applications, arXiv preprint arXiv:2402.00090 (2024).

[12] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Y. Wang, S. Riedel, Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Advances in Neural Information Processing Systems (NeurIPS), volume 33, 2020, pp. 9459–9474.

[13] J. Shang, T. Ma, C. Xiao, J. Sun, Gamenet: Graph augmented memory networks for recommending medication combination, in: Proc. AAAI Conf. Artif. Intell., volume 33, 2019, pp. 1126–1133.