Pacific Northwest
NATIONAL LABORATORY

PNNL-36792

Retrieval Augmented Generation for Robust Cyber Defense

September 2024

Moqsadur Rahman
Krish O Piryani
Aaron M Sanchez
Sai Munikoti
Luis De La Torre
Maxwell S Levin
Monika Akbar
Mahmud Hossain
Monowar Hasan
Mahantesh Halappanavar

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

PNNL-36792

Retrieval Augmented Generation for Robust Cyber Defense

September 2024

Moqsadur Rahman
Krish O Piryani
Aaron M Sanchez
Sai Munikoti
Luis De La Torre
Maxwell S Levin
Monika Akbar
Mahmud Hossain
Monowar Hasan
Mahantesh Halappanavar

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

In cybersecurity, the ability to efficiently analyze and respond to vulnerabilities, weaknesses, attack patterns, and threat tactics is critical for effective defense strategies. With the increasing complexity and volume of cybersecurity data, traditional methods of querying and retrieving information are often inadequate. To address this challenge, we implemented Retrieval-Augmented Generation (RAG) systems—CyRAG and GraphCyRAG—that integrate large language models (LLMs) with both structured data from relational databases and knowledge graphs such as Neo4j. CyRAG is designed to handle structured data, focusing on CVE (Common Vulnerabilities and Exposures) and CWE (Common Weakness Enumeration) entities to generate accurate and context-rich responses. In contrast, GraphCyRAG leverages Neo4j knowledge graphs to retrieve interconnected information from CVE, CWE, CAPEC (Common Attack Pattern Enumeration and Classification), and ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) datasets. By utilizing Neo4j's graph-based framework, GraphCyRAG enables deeper traversal of relationships between vulnerabilities and attack patterns, providing cybersecurity analysts with more comprehensive insights into potential attack vectors and mitigation strategies. Our preliminary results demonstrate that integrating knowledge graphs with RAG significantly enhances both the accuracy and depth of threat analysis, allowing for the retrieval of dynamic, real-time data and the generation of contextually aware responses. This approach helps analysts uncover hidden relationships between cyber entities, predict exploit paths, and prioritize mitigation efforts effectively. The integration of RAG with cybersecurity knowledge graphs represents a significant advancement in cybersecurity threat intelligence, enabling more informed decision-making and stronger defense strategies.

Summary

We developed two tools CyRAG and GraphCyRAG for enable efficient cyber defense. CyRAG brings together structured data from common vulnerabilities and exposures (CVE) and common weakness enumeration (CWE) to generate accurate and context-rich responses from a large language model using retrieval augmentation technique. GraphCyRAG employs Neo4j knowledge graphs to retrieve interconnected information from CVE, CWE, CAPEC (Common Attack Pattern Enumeration and Classification), and ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) datasets. Our early results demonstrate significant enrichment to both the accuracy and the depth of threat analysis using retrieval of dynamic, real-time data and the generation of contextually aware responses. Our work represents a significant advancement in cybersecurity threat intelligence, enabling informed decisions and optimized cyber defense.

Acknowledgments

This research was supported by the **Mathematics for Artificial Reasoning in Science Initiative**, under the Laboratory Directed Research and Development (LDRD) Program at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

Acronyms and Abbreviations

CVE: Common Vulnerabilities and Exposures

CWE: Common Weakness Enumeration

CAPEC: Common Attack Pattern Enumeration and Classification

ATT&CK: Adversarial Tactics, Techniques, and Common Knowledge

RAG: Retrieval Augmented Generation

LLM: Large Language Model

KG: Knowledge Graph

Contents

Figures

1.0 Introduction

In cybersecurity, understanding the intricate relationships between vulnerabilities, weaknesses, attack patterns, and known attacks is not only critical for effective threat mitigation and proactive defense strategies but also essential for comprehensive threat intelligence [1-9]. Large-scale cybersecurity databases, such as CVE (Common Vulnerabilities and Exposures), CWE (Common Weakness Enumeration), CAPEC (Common Attack Pattern Enumeration and Classification), and ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge), provide an immense wealth of structured data that can be harnessed for deep analysis and insights. However, these databases alone may not be enough to extract meaningful connections and actionable intelligence efficiently. This is where combining knowledge graphs and Retrieval-Augmented Generation (RAG) significantly enhances capabilities.

A network or graph, G=(V, E), is a pair consisting of a set of unique entities known as nodes or vertices (V) and binary relationships on these entities known as links or edges (E). A knowledge or semantic graph is a graph of real-world entities such as people, institutions, concepts or events, and their relationships. A knowledge graph, such as one built using Neo4j, offers an intuitive way to model relationships between different entities in the cybersecurity domain. These entities—vulnerabilities (CVE), weaknesses (CWE), attack patterns (CAPEC), and adversarial tactics (ATT&CK)—can be represented as nodes, with the relationships between them as edges. This interconnected structure helps security professionals visualize and analyze how vulnerabilities may lead to specific attacks, how weaknesses in software can be exploited, and how attack patterns can escalate into larger security threats. By mapping these relationships in a graph format, knowledge graphs make it easier to uncover hidden patterns, draw conclusions, and make connections that are difficult to observe in siloed datasets.

In a conversational (chatbot) context, responses from generative large language models (LLMs) suffer from several limitations: lack of support from authoritative sources for the response (explainability and trust), lack of temporal adjustment to factoids that change over time (drift), and responses lacking a confidence metric from the model (especially for queries for which no answers should be provided). Augmenting queries and responses with evidence from a carefully curated database can address many of these limitations. A retrieval-augmented generation (RAG) system builds a database by indexing documents (and parts of documents) using an LLM-based encodings, and then retrieves the most relevant documents (or parts of a document) for a given query. These documents are then used as a context along with the user provided query (prompt) to an LLM. The responses from an LLM are thus based not only on the prompt, but also based on the context provided by the curated database. Since retrieved documents are provided along with the responses from an LLM, the user can assess responses in the context of retrieved documents (evidence).

The integration of retrieval-augmented generation (RAG) with knowledge graphs further amplifies these capabilities. RAG is an advanced approach that leverages both information retrieval and generative models to produce accurate, context-aware, and

detailed responses to complex cybersecurity queries. Traditional query-response systems often rely on pre-trained models that may not have access to the most up-to-date or comprehensive information. RAG, in contrast, bridges this gap by retrieving relevant data from vast external sources, such as knowledge graphs, and using this retrieved information as a basis for generating informed responses. This makes the system highly dynamic and capable of dealing with real-time information and complex datasets.

The advantages of combining RAG and knowledge graphs:

- **Real-Time and Context-Aware Responses**: The retrieval aspect of RAG allows the system to access the most relevant and current information stored in the knowledge graph. For example, when a user queries a specific CVE, the system can traverse through the graph to identify associated CWEs, CAPECs, and ATTACK patterns in real-time. This ensures that the generated responses are not only accurate but also context-aware, reflecting the latest known relationships and risks.
- **Enhanced Threat Intelligence:** Knowledge graphs represent relationships between vulnerabilities and attacks in a structured way, making it easier to uncover complex attack paths. With RAG, these paths can be retrieved and interpreted efficiently, allowing security professionals to assess the full impact of a vulnerability. For instance, understanding how a specific CWE can lead to an attack pattern (CAPEC) that is linked to real-world attack techniques (ATT&CK) helps in prioritizing mitigations and defending against potential threats.
- **Improved Decision-Making:** Security analysts can make better decisions when they have access to both the granular details of vulnerabilities and a high-level overview of how different cybersecurity entities are interconnected. RAG retrieves relevant data from the knowledge graph and generates detailed reports or summaries, which helps analysts prioritize vulnerabilities, predict possible exploitation paths, and implement effective mitigations.
- **Scalability and Flexibility:** Cybersecurity threats evolve rapidly, and the datasets are continuously growing. Knowledge graphs offer a scalable and flexible solution to handle this complexity. By integrating RAG, the system can adapt to changing datasets and still provide precise answers by dynamically retrieving and generating responses based on the most up-to-date information.
- **Automation of Complex Queries:** Without RAG and knowledge graphs, security analysts might need to manually explore vulnerabilities, attack patterns, and mitigation strategies, which can be time-consuming and error prone. RAG automates this process by querying the knowledge graph and generating insights quickly, freeing up valuable time for analysts to focus on more strategic tasks.

In our implementation, we have developed two distinct types of RAG systems for cybersecurity: CyRAG and GraphCyRAG.

- **CyRAG** is a traditional RAG system that focuses on structured retrieval from large cybersecurity databases like CVE and CWE. It enables users to access detailed information about specific vulnerabilities and weaknesses and uses that information

to generate comprehensive, detailed reports. This system is particularly useful for analysts looking for specific information about vulnerabilities and their associated risks.

- **GraphCyRAG**, however, takes a more interconnected approach by leveraging graph-based data from Neo4j. This system is designed to explore the complex relationships between vulnerabilities, weaknesses, attack patterns, and known attacks. By traversing through the graph, it provides a richer and more comprehensive analysis of cyber threats, making it ideal for in-depth threat modeling and analysis.

By combining the strengths of knowledge graphs and RAG, our systems—CyRAG and GraphCyRAG—offer security professionals a powerful toolset for understanding, analyzing, and defending against the ever-evolving landscape of cyber threats. Through real-time data retrieval and context-aware generation, these systems enable deeper insights and actionable intelligence, and significantly enhancing how cybersecurity data is accessed and utilized for effective threat mitigation, defense strategies, and decision-making.

In the following sections, we detail our work on building the GraphCyRAG tool and demonstrate the potential benefits of this approach.

2.0 Knowledge Graph Encoding of Cyber Information

MITRE information used to build the knowledge graph:

- **CVE (Common Vulnerabilities and Exposures):** CVE is a catalog of known security vulnerabilities in software. Each vulnerability is assigned a unique identifier (CVE ID), which helps in tracking, discussing, and addressing these issues. When a security flaw is discovered in a program that could be exploited by attackers, it is documented in the CVE system so that organizations can take necessary actions to protect their systems.
- **CWE (Common Weakness Enumeration):** CWE is a catalog of common weaknesses in software that can lead to security vulnerabilities. It serves as a reference for identifying poor coding practices that might expose software to attacks. By understanding these weaknesses, developers can avoid them during the coding process, ultimately creating more secure software.
- **CAPEC (Common Attack Pattern Enumeration and Classification):** CAPEC is a catalog of common attack patterns used by attackers to compromise systems. It outlines various strategies and techniques that hackers employ. By studying these attack patterns, security professionals can strengthen defenses and better predict and prevent potential threats.
- **ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge):** MITRE ATT&CK is a framework that details the behaviors and techniques attackers use once they've gained access to a system. It serves as a guide, outlining the steps attackers take after breaking in. This information helps defenders identify and counteract malicious activities before they can cause significant damage.
- **Mitigations:** Mitigations are measures or controls designed to reduce or prevent damage from security threats. These can include implementing best practices, applying software updates, and using security tools. By putting the right mitigations in place, organizations can safeguard their systems against known vulnerabilities and attack methods.

Knowledge graph schema: Four different types of nodes and relationships are used to build the knowledge graph:

Nodes: We consider a total of 245,125 number of nodes in this work:

- **CVE:** CVE nodes represent known security vulnerabilities within the knowledge graph. These nodes contain key properties that describe the nature, severity, and impact of the vulnerability. There are 242504 CVE nodes. Below are the properties of a CVE node:

    - id
    - name
    - description
    - base_score
    - category

- exploitability_score
- impact_score
- related_cwe_ids
- severity
- embedding

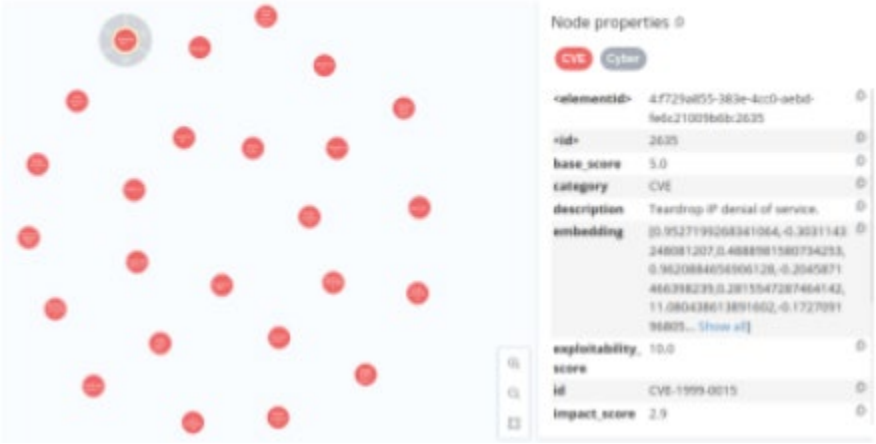

Figure 1. CVE nodes represent known security vulnerabilities within the knowledge graph

- **CWE:** CWE nodes represent common weaknesses in software within the knowledge graph. These nodes contain properties that describe the nature and relationships of these weaknesses. In total there are 963 CWE nodes. Below are the properties of a CWE node:

  - category
  - child_of_ids
  - description
  - embedding
  - id
  - name

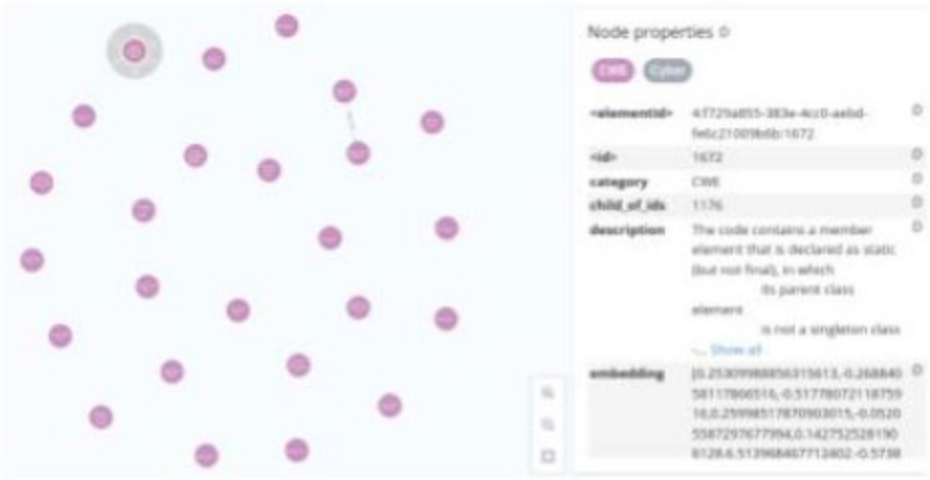

Figure 2. CWE nodes represent common weaknesses in software within the knowledge graph.

- **CAPEC:** CAPEC nodes represent common attack patterns within the knowledge graph. These nodes include properties that describe the characteristics, relationships, and impact of the attack patterns. There are 615 CAPEC nodes. Below are the properties of a CAPEC node:

    - category
    - childOf
    - description
    - embedding
    - id
    - likelihood
    - mitigations
    - name
    - severity

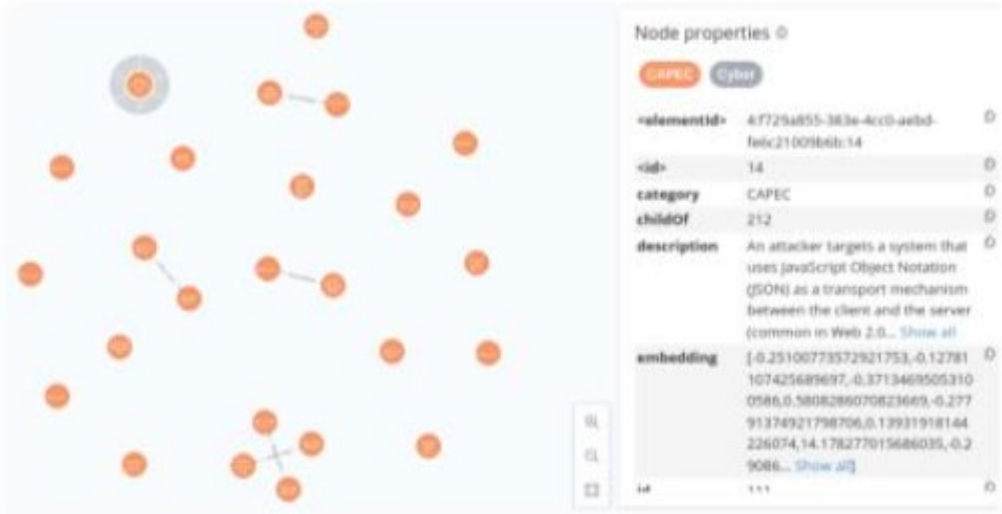

Figure 3. CAPEC Nodes represent common attack patterns within the knowledge graph.

- **ATTACK:** The attack node represents a critical element that encapsulates detailed information about specific cyber threats or malicious activities. The structure of the Attack node is built upon a set of well-defined properties, which provide a comprehensive understanding of the attack's characteristics and context. There are 1,043 ATTACK nodes. Here are the ATTACK node properties:

    - id
    - name
    - description
    - type
    - category
    - embedding

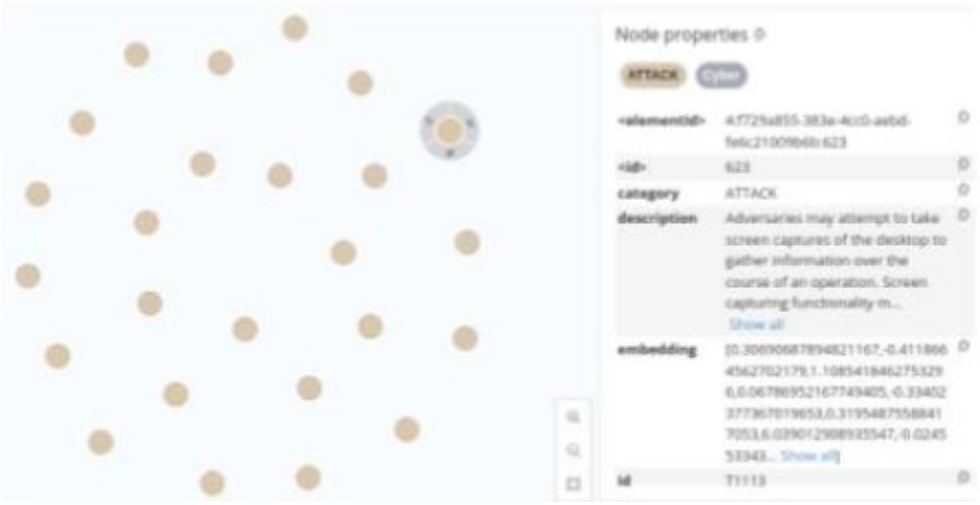

Figure 4. Attack Nodes represents a critical element that encapsulates detailed information about specific cyber threats or malicious activities within the knowledge graph.

Relationships: There are a total of 270,093 relationships:

- **CVE_CWE:** This relationship connects a CVE node to a CWE node, indicating that a specific vulnerability (CVE) is associated with a particular weakness (CWE) in software. It helps in understanding which weaknesses lead to specific vulnerabilities. In total, there are 260888 numbers of CVE_CWE relationships.

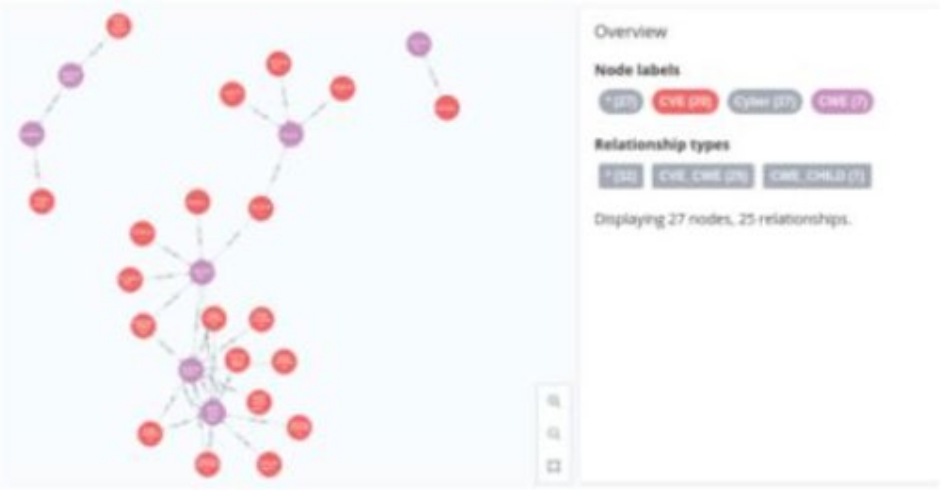

Figure 5. The relationship that connects a CVE node to a CWE node, indicating that a specific vulnerability (CVE) is associated with a particular weakness (CWE) in software within the knowledge graph.

- **CWE_CHILD:** This relationship links a CWE node to another CWE node, representing a hierarchical relationship where one weakness (child CWE) is a more specific instance or a subset of a broader weakness (parent CWE). There are 1,307 CWE_CHILD relationships.

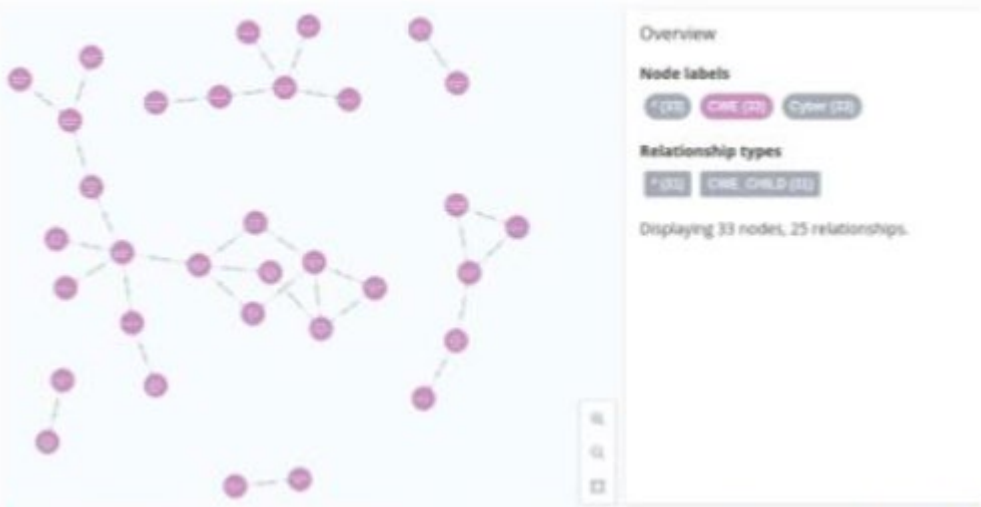

Figure 6. CWE node and CWE node relationship where one weakness (child CWE) is a subset of broader weakness (parent CWE) within the knowledge graph.

- **CWE_CAPEC:** This relationship connects a CWE node to a CAPEC node, showing that a particular software weakness (CWE) can be exploited using a specific attack pattern (CAPEC). It helps map weaknesses to potential attack strategies. There are 6,931 relationships of type CWE_CAPEC.

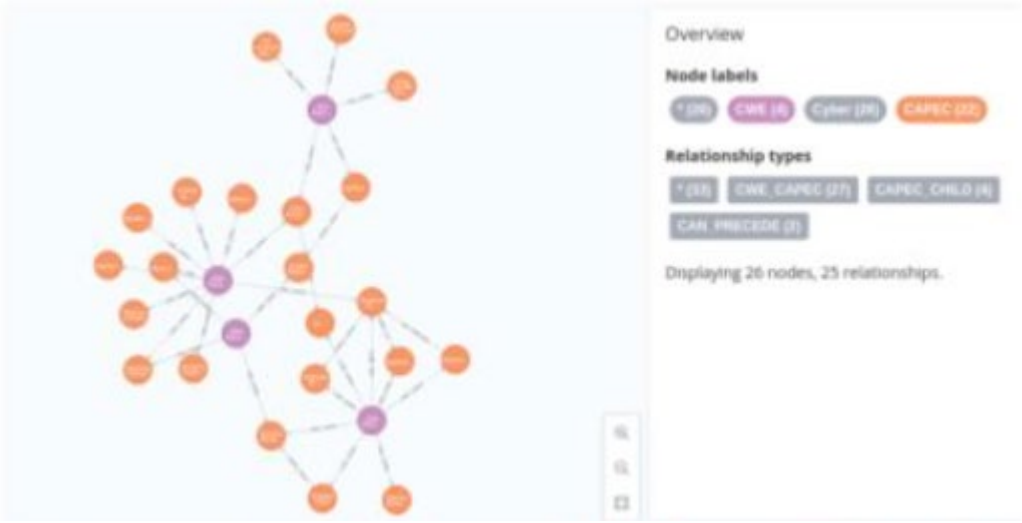

Figure 7. The relationship that connects CWE node to CAPEC node within the knowledge graph.

- **CAPEC_CHILD:** This relationship links a CAPEC node to another CAPEC node, indicating a hierarchical relationship where one attack pattern (child CAPEC) is a more specific variation or part of a broader attack pattern (parent CAPEC). There are 533 CAPEC_CHILD relationships.

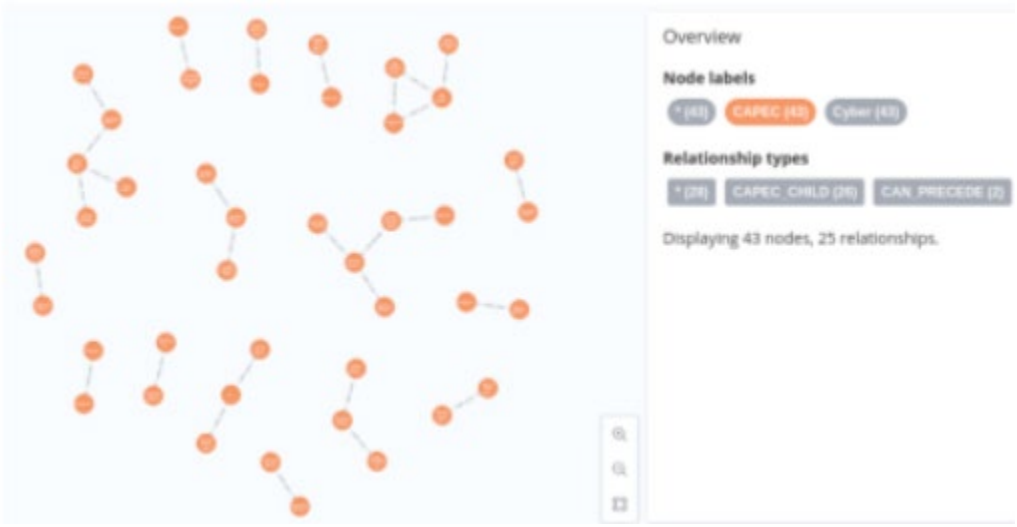

Figure 8. The hierarchical relationship which a CAPEC node (child CAPEC) is a more specific variation or part of a broader attach pattern (parent CAPEC) within the knowledge graph.

- **CAN_PRECEDE:** This relationship between two CAPEC nodes indicates that one attack pattern (CAPEC) can be a precursor or a preceding step to another attack pattern. It helps in understanding the sequence or progression of attack techniques. The total number of CAN_PRECEDE relationships is 162.

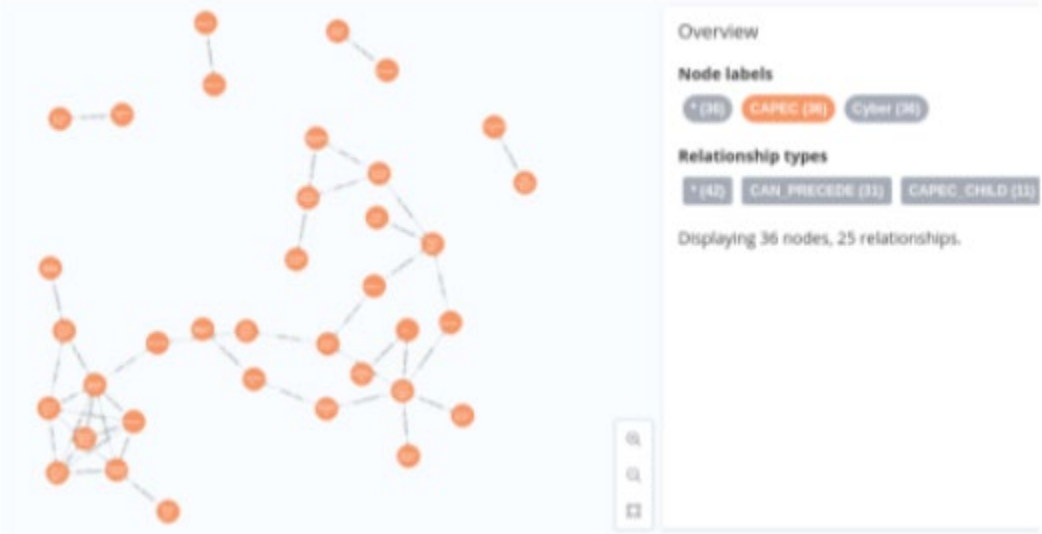

Figure 9. The relationship between two CAPEC nodes indicates that one attack pattern (CAPEC) can be a precursor or a preceding step to another pattern within a knowledge graph.

- CAPEC_ATTACK: This relationship connects a CAPEC node to an Attack node, showing that a specific attack pattern (CAPEC) is used in or is related to a particular attack. It helps in associating specific tactics and techniques with known attacks in the system. A total of 272 relationships exists under the CWE_CAPEC category.

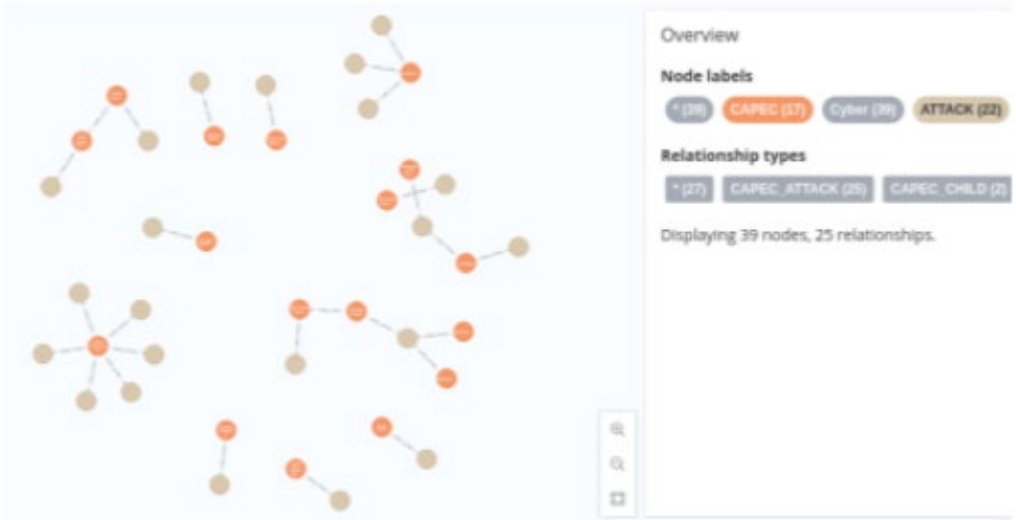

Figure 10. The relationship between a CAPEC node to an attack node, showing that a specific attack pattern (CAPEC) is used in or is related to a particular attack in a knowledge graph.

+Neo4j Knowledge Graph Implementation: Implementing a Neo4j Knowledge Graph for cybersecurity requires several steps to manage and analyze interconnected data effectively. The process includes data collection, preprocessing, defining entities, relationships, and properties, and finally importing the data into Neo4j.

- **Data Collection:** The first step in the Neo4j Knowledge Graph implementation is gathering relevant cybersecurity data from authoritative sources. Vulnerabilities (CVEs) are sourced from the National Vulnerability Database (NVD), which provides detailed information about known vulnerabilities in software and systems. Weaknesses (CWEs), attack patterns (CAPECs), and adversarial tactics/techniques (ATT&CK) are obtained from MITRE, a globally recognized organization that maintains comprehensive cybersecurity frameworks. Collecting this data is critical for building a rich, interconnected knowledge graph that captures the relationships between these cybersecurity concepts, forming the foundation for further processing and analysis.
- **Preprocessing:** After the data is collected, preprocessing is required to clean and structure the information to ensure compatibility with Neo4j. This includes removing or replacing unallowed characters that could interfere with database operations and formatting the data according to the requirements of the graph database. The goal of preprocessing is to make the raw data usable by converting it into a clean, structured format that is optimized for efficient querying and relationship mapping within the Neo4j database.
- **Defining Entities,** Properties, and Relationships: The next step is defining the entities, properties, and relationships that will structure the knowledge graph. Entities such as vulnerabilities (CVEs), weaknesses (CWEs), attack patterns (CAPECs), and adversarial tactics/techniques (ATT&CK) are modeled as nodes within the graph. Each entity is associated with specific properties that describe its attributes, such as CVE ID, description, impact score, and exploitability. Relationships between these entities are established as edges, illustrating how attack patterns exploit weaknesses or how specific vulnerabilities map to tactics and techniques. These relationships also carry properties, such as severity or likelihood, to capture the context of these interactions in the cybersecurity landscape.
- **Data Import into Neo4j:** Once the entities, properties, and relationships are defined, the data is ready to be imported into Neo4j. Using Cypher queries or batch import methods, the preprocessed data is loaded into the Neo4j database, establishing the graph structure. This step involves mapping the structured data into the defined nodes and edges, ensuring that the relationships between entities are accurately reflected in the graph.

The Neo4j Knowledge Graph thus enables cybersecurity practitioners to gain deeper insights, identify patterns, and analyze threats more effectively, making it an essential tool for threat intelligence, vulnerability management, and incident response.

Cypher Queries: Cypher, Neo4j's query language, provides an intuitive and powerful way to retrieve insights from graph databases. Whether performing basic queries or tackling more advanced data retrieval tasks, Cypher excels in handling complex

relationships and patterns in large datasets. For instance, it can be used to count specific types of nodes or relationships, explore relationships between different entities, or identify connections between cybersecurity vulnerabilities and threats. This makes Neo4j an indispensable tool for advanced data analysis and threat intelligence, particularly in domains like cybersecurity where understanding interconnected data is crucial. Here are some examples:

- **Counting Nodes:** Cypher can count the number of specific types of nodes in a Neo4j database, providing insights into the composition of the graph. For example, in a cybersecurity knowledge graph, one might want to count how many nodes represent vulnerabilities, weaknesses, attack patterns, or attack. This is essential for understanding the distribution and prevalence of different entities in the dataset.

Query:

```
MATCH (n:CVE)
RETURN COUNT(n) AS cve_count;
```

Response:

```
242504
```

Figure 11. Example of how a Cypher query can count the number of specific types of nodes in a Neo4j database, providing insights into the composition of the graph.

- **Relationships:** Cypher can count the number of specific types of relationships between nodes, such as finding how many “CWE_CAPEC” relationships exist in a cybersecurity graph. This provides valuable insights into the structure and frequency of certain relationships within the data.

Query:

```
MATCH ()-[r:CWE_CAPEC]->()
RETURN COUNT® AS relationship_count;
```

Response:

```
6931
```

Figure 12. Example of how a Cypher query can count the number of specific types of relationships between nodes.

- Node Information Extraction: Cypher allows for extracting detailed information about specific nodes in a Neo4j database. In a cybersecurity knowledge graph, where each node might represent a vulnerability, weakness, or other entities, retrieving the properties of a node can provide essential insights into its characteristics and associated metadata. For instance, if one wants to explore the properties of a particular vulnerability, they can query for the node with a specific CVE_ID and retrieve

all its associated properties. This is particularly useful for analyzing the attributes of vulnerabilities, understanding their relationships to other entities, and gaining a deeper understanding of the dataset.

Figure 13. Example of a cypher query for the node with a specific CVE_ID and retrieve all its associated properties to explore the properties of a particular vulnerability within a knowledge graph.

- **Exploring Cybersecurity Relationships:** Exploring Cybersecurity Relationships with Cypher Queries In the world of cybersecurity, understanding the connections between various types of cyber nodes is critical for identifying potential risks, attack vectors, and mitigation strategies. Neo4j's Cypher query language allows analysts to efficiently navigate between different entities, such as vulnerabilities (CVE), weaknesses (CWE), attack patterns (CAPEC), and specific attack techniques (ATTACK), providing a comprehensive view of how these elements are related.
- pher queries can be designed to trace the shortest or most relevant paths between these nodes by following predefined relationships such as CVE_CWE (linking a vulnerability to a weakness), CWE_CAPEC (connecting weaknesses to attack patterns), and CAPEC_CHILD (defining hierarchical relationships between attack patterns). Through these traversals, analysts can quickly uncover the relationships between seemingly disparate entities, identifying patterns that might not be obvious from a single node perspective.
- For example, by linking a specific CVE to its related CWEs and following the chain to associated CAPEC attack patterns, security professionals can predict how a vulnerability could be exploited in real-world attack scenarios. This helps them assess the broader impact of vulnerabilities, prioritize patches or mitigations, and understand the potential weaknesses that an attacker might exploit.
- Moreover, Cypher queries enable deeper threat modeling by providing a visual map of how various cyber elements are interconnected. This holistic view is invaluable for

cybersecurity professionals, as it reveals not only individual threats but also how multiple vulnerabilities and weaknesses might work together in an attack chain. With this knowledge, teams canbuild more robust defense strategies, focusing resources on high-risk areas and understanding the full scope of potential cyber threats.

Example 1: Finding CAPECs from a given CVE:

Query:

```
MATCH (cve:CVE {id: 'CVE-2021-34527'})
MATCH path = (cve)-[:CVE_CWE*0..2]->(cwe:CWE)-[:CWE_CAPEC*0..2]-
>(capec:CAPEC)
RETURN path, length(path) AS path_length, nodes(path), relationships(path)
ORDER BY path_length ASC
LIMIT 10
```

Response:

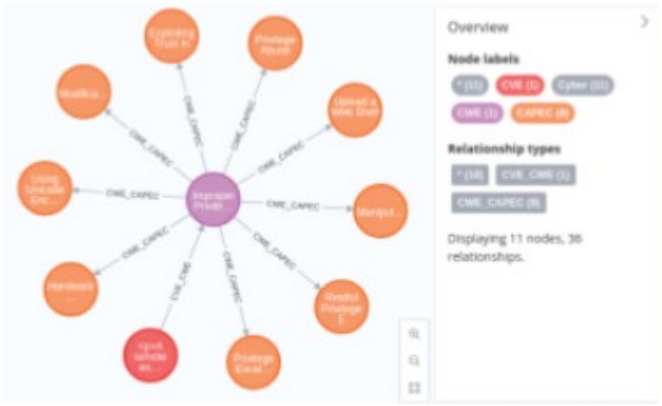

Figure 14. Finding CAPECs from a given CVE

Example 2: Finding ATTACKs from a given CVE:

Query:

```
MATCH (cve:CVE {id: 'CVE-2021-34527'})
MATCH path = (cve)-[:CVE_CWE*0..2]->(cwe:CWE)-[:CWE_CAPEC*0..2]-
>(capec:CAPEC)-[:CAPEC_CHILD|CAN_PRECEDE*0..2]->(capec2:CAPEC)-
[:CAPEC_ATTACK*0..2]->(attack:ATTACK)
RETURN path, length(path) AS path_length, nodes(path), relationships(path)
ORDER BY path_length ASC
LIMIT 10
```

Response:

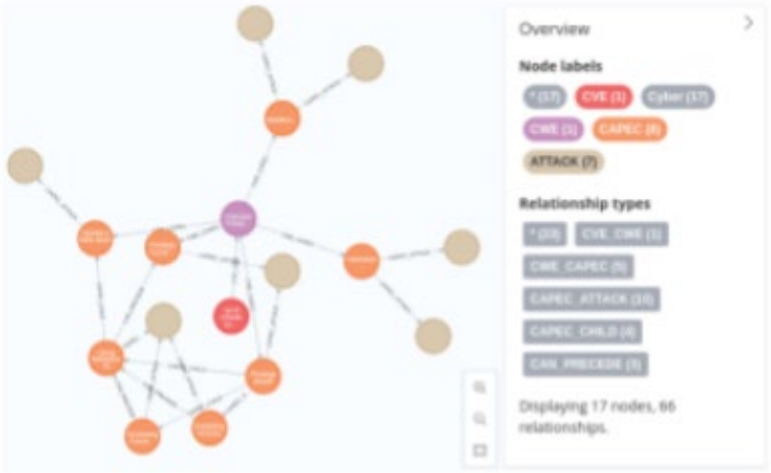

Figure 15. Finding ATTACKs from a given CVE.

3.0 Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) is an innovative method in natural language processing that enhances the capabilities of generative models by incorporating information retrieval. RAG combines two powerful elements: retrieving relevant data from external sources and generating a coherent, contextually accurate response based on that retrieved information. This approach addresses the limitations of generative models that rely solely on pre-trained knowledge and enhances their ability to provide more accurate and detailed answers.

Key Concepts:

- Retrieval: The model retrieves relevant documents or data from external sources, such as a database, search index, or knowledge base. This step allows the model to access up-to-date specific information that might not be contained in the generative model's training data.
- Generation: After retrieving the relevant data, the generative model uses it to formulate a response. By integrating this external information, the model generates outputs that are more context-aware, accurate, and specific to the query at hand.

How RAG Works:

- Input: A user query or input is provided to the model.
- Information Retrieval: The model searches through a large collection of documents or data sources, fetching the most relevant pieces of information.
- Generation: The retrieved information is fed into a generative model, such as a transformer-based model, which uses it to produce a coherent and contextually enriched response.

General Applications of RAG:

- Customer Support: RAG can improve automated customer support systems by retrieving the most relevant information from a company's knowledge base to answer specific customer queries.
- Research and Knowledge Discovery: Researchers can use RAG to retrieve and summarize relevant literature, allowing them to gain insights from vast collections of research papers or data sets.
- Chatbots and Assistants: RAG enhances virtual assistants and chatbots, enabling them to provide more accurate, fact-based answers by retrieving real-time information from the web or internal databases.

We have built two distinct types of Retrieval-Augmented Generation (RAG) systems specifically designed for cybersecurity entities: CyRAG and GraphCyRAG. Both models leverage different methods to enhance how information related to cybersecurity vulnerabilities and attack patterns is retrieved and generated, but they serve slightly different purposes based on their underlying structures.

CyRAG:

CyRAG is a traditional Retrieval-Augmented Generation system built to handle structured data from cybersecurity databases, specifically focusing on CVE (Common Vulnerabilities and Exposures) and CWE (Common Weakness Enumeration) entities. It allows users to retrieve detailed information about vulnerabilities, weaknesses, and exploit patterns, and uses that information to generate detailed reports, explanations, or summaries of cyber threats. For example, when a user queries CyRAG with a specific CVE ID, the system retrieves the most relevant information from a CVE database (including associated CWEs, impacts, and mitigations) and uses this information to generate a comprehensive response. CyRAG is ideal for analysts who need fast, accurate, and detailed insights into vulnerabilities and their corresponding weaknesses.

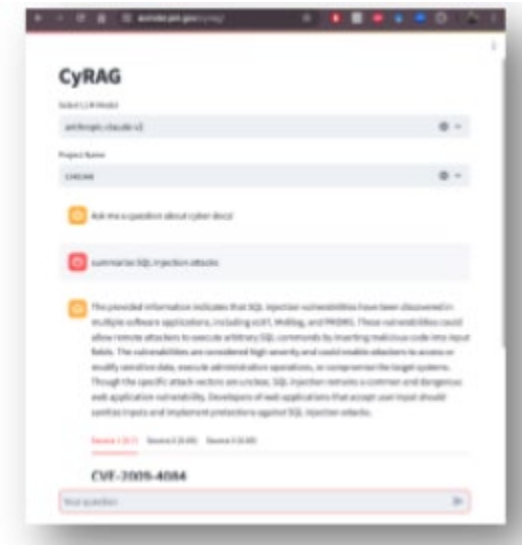

Figure 16. Example: Upon receiving a query with a specific CVE ID, CyRAG retrieves relevant data from the CVE database, including associated CWEs, impacts, and mitigations, to generate a comprehensive response.

GraphCyRAG:

Graph Retrieval-Augmented Generation (GraphCyRAG) is an advanced approach that combines the power of graph databases, like Neo4j, with large language models (LLMs) to generate precise, data-driven responses. By leveraging structured data from the graph, this method enhances the accuracy and context of the generated outputs. In cybersecurity, this can be particularly useful when analyzing relationships between vulnerabilities (CVE), weaknesses (CWE), attack patterns (CAPEC), and actual attacks (ATTACK). Here's a breakdown of how our GraphCyRAG works:

- Converting User Query to Cypher Query: When a user asks a question, the system first translates it into a Cypher query—Neo4j's native query language. The purpose of this step is to interact with the knowledge graph and fetch the most relevant data points.

Example: Suppose a user queries, “What are the possible attacks related to CWE 89?”

```
MATCH (cwe:CWE {id: '89'})
MATCH path = (cwe)-[:CWE_CAPEC*0..2]->(capec:CAPEC)-
[:CAPEC_CHILD|CAN_PRECEDE*0..2]->(capec2:CAPEC)-[:CAPEC_ATTACK*0..2]-
>(attack:ATTACK)
RETURN path, length(path) AS path_length, nodes(path), relationships(path)
ORDER BY path_length ASC
LIMIT 10
```

Figure 17. Cypher query results from a user query: “What are the possible attacks related to CWE 89?”

This query navigates the knowledge graph, searching for connections between the CWE, CAPEC, and ATTACK nodes to find potential attack patterns linked to the given vulnerability.

- **Executing Cypher Query on Neo4j Knowledge Graph:** Once the query is generated, it is executed against the Neo4j knowledge graph, which contains structured data on vulnerabilities, weaknesses, attack patterns, and attack techniques. This graph might have thousands of nodes and relationships, but the Cypher query filters and fetches the most relevant nodes and relationships based on the user's input.

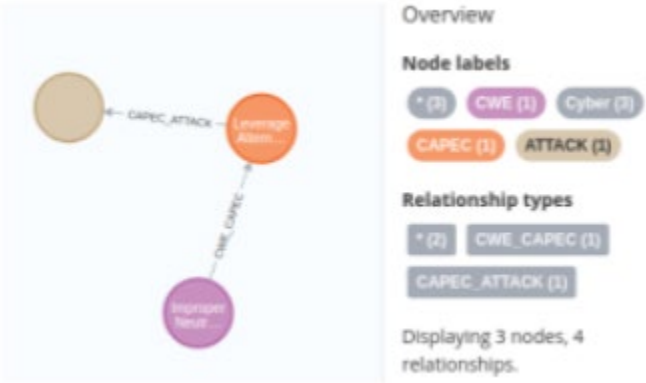

Figure 18. For the query about CWE 89, Neo4j retrieves a list of connected entities – CWE, CAPEC, and ATTACK nodes – that show how this vulnerability can potentially be exploited or mitigated.

- **Preprocessing the Cypher Query Output:** After the data is retrieved from the knowledge graph, it needs to be preprocessed before being fed into the LLM. This step involves cleaning the data, structuring it for easy interpretation, and ensuring that only the most relevant information is passed to the LLM.

Example: When querying a Neo4j knowledge graph for information related to a specific CWE node, such as CWE-89 (SQL Injection), the Cypher query may retrieve related nodes up to the ATTACK level. However, the raw output might include extraneous details that are unnecessary for Large Language Models (LLMs) to generate responses, such as embedding vectors. Here's an example of how the output might be processed to remove such unnecessary data and focus on the critical relationships.

For example, the query returns related nodes, including CWE, CAPEC, and ATTACK nodes, along with embedding vectors that are not required for response generation.

```
{
  "nodes": [
    {"id": "CWE-89", "name": "SQL Injection", "embedding": [0.12, 0.45,
0.67, 0.22, .....]},
    {"id": "CAPEC-123", "name": "Injection Attack", "embedding": [0.75,
0.22, 0.33, 0.88, .....]},
    {"id": "T1134", "name": "Privilege Escalation", "embedding": [0.41,
0.68, 0.57, 0.12, .....]}
  ],
}

During post-processing, we focus on stripping away unnecessary data such
as embedding vectors, leaving only the relevant node attributes like id,
name, etc.:
{
  "nodes": [
    {"id": "CWE-89", "name": "SQL Injection"},
    {"id": "CAPEC-123", "name": "Injection Attack"},
    {"id": "T1134", "name": "Privilege Escalation"}
  ],
}
```

Figure 19. The query results contain related nodes, including CWE, CAPEC, and ATTACK, along with embedding vectors that are not required to response generation.

- **LLM-Based Response Generation:** The cleaned and structured data is then passed to the language model (LLM), which uses its generative abilities to produce a coherent, context-aware response. It can answer the questions that regular LLMs, or RAG Systems cannot answer precisely. For our experiments, we have used llama3:

  Example 1: User query - “How many CVEs are there? Give me the approximate count.”

Regular LLM's Response:

```
As of October 2021, there were over 180,000 CVEs (Common
Vulnerabilities and Exposures) listed in the National Vulnerability
Database (NVD). However, this number is constantly growing as new
vulnerabilities are discovered and assigned CVEs, so the approximate
count of CVEs is always changing.
```

GraphCyRAG Response:

```
The approximate count of CVEs in the graph database is 242,504.
```

Figure 20. Regular LLM's Response compared to GraphCyRag Response to User Query: "How many CVEs are there? Give me the approximate count."

Observation: GraphCyRAG responded with updated and precise information.

Example 2: User query – "Tell me about CWE 152 and its respective mitre CAPEC and ATT&CK techniques."

Regular LLM's Response:

```
CWE 152, also known as "Improper Neutralization of Input During Web
Page Generation (Cross-site Scripting)," is a common vulnerability in
web applications that allows attackers to inject malicious code into a
webpage. This code can then be executed by unsuspecting users, leading
to potential security breaches. The weakness is categorized under
input validation vulnerabilities, as it stems from improper handling
of user input.

This vulnerability is listed in the Common Weakness Enumeration (CWE)
database, which is a widely recognized list of common software
security flaws. CWE 152 is also associated with the Common Attack
Pattern Enumeration and Classification (CAPEC) database, which
outlines known attack patterns and helps in developing security
defenses.

In the MITRE ATT&CK framework, CWE 152 is linked to adversary tactics
and techniques like "Scripting" and "Web Shell" under the "Execution"
tactic, and "Exploit Public-Facing Application" under the "Initial
Access" tactic. These techniques highlight the potential for this
vulnerability to be exploited in significant security breaches within
web applications.
```

GraphCyRAG Response:

```
CWE 152, also known as "Improper Neutralization of Macro Symbols," is
a software vulnerability that occurs when a system improperly handles
macro symbols, which can lead to security issues such as code
injection or privilege escalation. This happens when a product
receives input but fails to correctly neutralize special elements that
are interpreted as macro symbols by downstream components.

CWE 152 is associated with CAPEC ID 13, "Subverting Environment
Variable Values." This technique allows an adversary to modify
environment variables in a way that disrupts the target software's
expected operation, potentially benefiting the attacker.

In the MITRE ATT&CK framework, CWE 152 is linked to techniques under
the category of "Obfuscated Files or Information." These techniques
involve the use of methods like encryption or encoding to obscure
files and evade detection. Specific techniques include T1574.006,
"Dynamic Linker Hijacking," and T1574.007, "Path Interception," which
attackers use to obfuscate actions and evade detection during various
stages of an attack.
```

Figure 21. Regular LLM's Response compared to GraphCyRag Response to: "Tell me about CWE 152 and its respective mitre CAPEC and ATT&CK techniques."

Observation: GraphCyRAG was able to find CWE 52 and its related CAPEC and ATT&CK information more precisely and accurately by utilizing the knowledge graph, whereas a regular LLM struggled to locate the corresponding CAPEC and provided less precise information regarding ATT&CK.

Example 3: User query – "Find the CAPECs related to CVE-2021-34527."

Regular LLM's Response:

```
Currently, there are no CAPECs (Common Attack Pattern Enumeration and
Classification) associated with CVE-2021-34527. CAPECs are usually
developed after a vulnerability has been identified and thoroughly
analyzed. It is possible that CAPECs related to this CVE might be
created in the future, but as of now, no known CAPECs are linked to
CVE-2021-34527.
```

GraphCyRAG Response:

```
CAPEC ID: 22
Name: Exploiting Trust in Client
Description: This attack type exploits vulnerabilities in the
authentication and data integrity of the client/server communication
channel. It takes advantage of the implicit trust a server places in
what it believes to be a legitimate client. An attacker carries out
this attack by communicating directly with the server, which
mistakenly believes it is interacting with a valid client. There are
several variations of this attack.

CAPEC ID: 122
Name: Privilege Abuse
Description: An adversary exploits features intended for privileged
users or administrators but are exposed to lower or non-privileged
accounts. Proper access control is crucial to ensure that only
authorized users can access sensitive information and functionality.

CAPEC ID: 76
Name: Manipulating Web Input to File System Calls
Description: An attacker manipulates inputs to the target software
that are passed to file system calls in the operating system. The goal
is to gain access to or modify areas of the file system that the
target software did not intend to make accessible.
```

Figure 22. Regular LLM's Response compared to GraphCyRag Response to: User query – "Find the CAPECs related to CVE-2021-34527."

Observation: GraphCYRAG was able to accurately retrieve related CAPECs for CVE-2021-345247, providing detailed and relevant information using the knowledge graph, while the regular LLM incorrectly stated that no CAPECs were associated with this CVE. The GraphCyRAG approach demonstrated higher precision by identifying specific CAPEC IDs and providing thorough descriptions of the related attack patterns, highlighting its advantage in utilizing structured data for better accuracy.

Example 4: User query – "Tell me the impact and exploitability score of CVE-1999-0009"

Regular LLM's Response

```
CVE-1999-0009 is a vulnerability in the Microsoft Windows NT operating
system that allows a remote attacker to cause a denial of service
(DoS) by sending a large number of specially crafted packets to the
affected system. The impact of this vulnerability is considered
moderate, as it can cause the system to become unresponsive or
potentially crash, but it does not allow for remote code execution or
access to sensitive information.
```

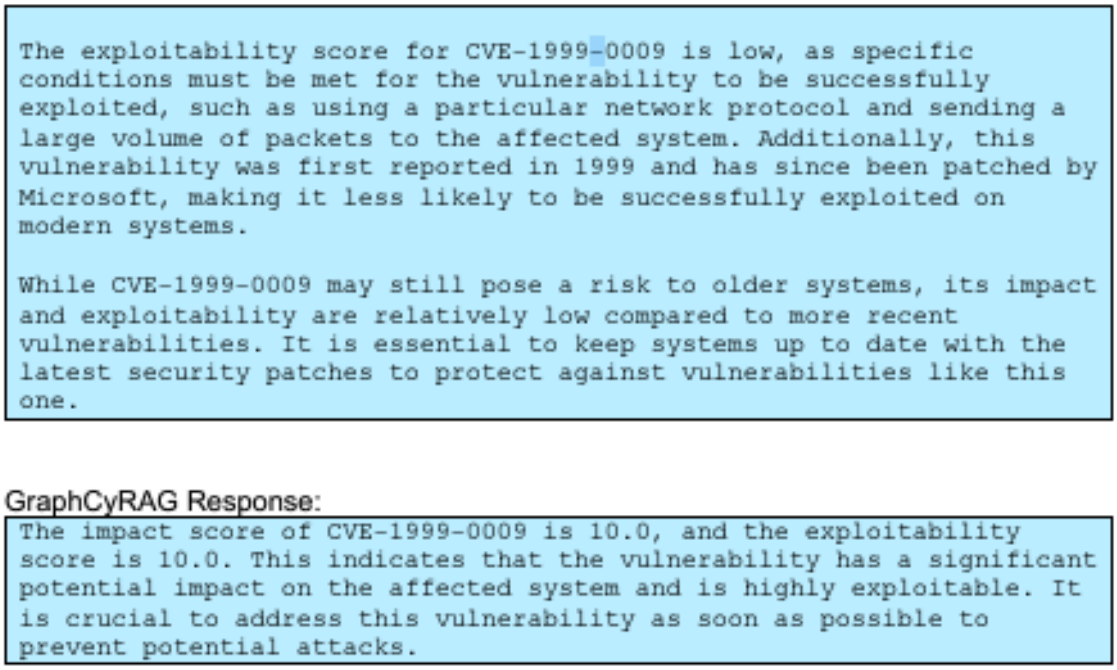

Figure 23. Regular LLMs Response compared to GraphCyRAG Response to: User query – "Tell me the impact and exploitability score of CVE-1999-0009".

Observation: The GraphCyRAG system provided a concise and accurate response by directly referencing the impact and exploitability scores of CVE-1999-0009, stating both as 10.0, and emphasizing the severity and urgency of addressing the vulnerability. In contrast , the regular LLM's response was more descriptive but lacked specific scores and provided a broader explanation regarding the conditions and history of the vulnerability. The GraphCyRAG system demonstrated greater precision by offering the exact scores relevant to the query, while the regular LLM offered a less focused and detailed answer.

4.0 Conclusions and Future Work

The preliminary results of integrating Retrieval-Augmented Generation (RAG) with knowledge graphs in cybersecurity demonstrate a significant enhancement in the ability to retrieve, analyze, and generate context-aware insights based on complex relationships between vulnerabilities, weaknesses, attack patterns, and threat tactics.

- Improved Accuracy in Query Responses: One of the key advantages of the RAG system, particularly GraphCyRAG, is its ability to generate highly accurate responses by leveraging both the structured data in knowledge graphs (such as Neo4j) and the generative capabilities of language models. Initial tests show that the retrieval step helps the generative model access up-to-date, relevant information, leading to more precise and context-rich answers.
- Efficiency in Data Retrieval: RAG's ability to combine retrieval and generation shows promising results in terms of efficiency. The preliminary analysis indicates that the use of a graph database like Neo4j, coupled with a retrieval mechanism, ensures that the system doesn't rely solely on the generative model's static knowledge. Instead, by accessing dynamic, real-time data from the graph, the system retrieves the latest information on vulnerabilities, weaknesses, and attack techniques. This results in faster and more efficient query processing, even when dealing with complex cybersecurity relationships.
- Deeper Insights with GraphCyRAG: GraphCyRAG shows potential in its ability to analyze and traverse relationships in a structured knowledge graph. Initial tests demonstrate that it can follow multiple relationship types (e.g., CVE_CWE, CWE_CAPEC) to reveal deeper insights into how specific vulnerabilities (CVEs) connect to weaknesses (CWEs) and attack patterns (CAPECs). This deeper understanding allows analysts to explore not just individual threats but also how these threats interact in real-world attack scenarios.

The preliminary results of the CyRAG and GraphCyRAG systems highlight the potential of combining knowledge graphs with Retrieval-Augmented Generation to enhance cybersecurity threat analysis. By integrating real-time data retrieval from a structured graph with the generative capabilities of modern language models, the system offers deeper insights, more accurate responses, and efficient data retrieval. This approach promises to revolutionize how cybersecurity data is processed, enabling analysts to make more informed, timely decisions.

In our future work, we plan to integrate RAG and GraphCyRAG into the production cyber operations tools at PNNL so that analysis can be performed directly on PNNL-specific systems and their vulnerabilities. Our goal is to enable the PNNL cyber operations team to enhance cyber security. We will then generalize the lessons learnt from this exercise to provide open source tools for the community at large.

5.0 References

1) S Das, E. Serra, M. Halappanavar, A. Pothen, and E. Al-Shaer. "V2W-BERT: A Framework for Effective Hierarchical Multiclass Classification of Software Vulnerabilities." In proceedings of the 8th IEEE International Conference on Data Science and Advanced Analytics (DSAA). Porto, Portugal. October 2021. **[Best Application Paper Award]**
2) "VWC-BERT: Scaling Vulnerability–Weakness–Exploit Mapping 59 60 3 on Modern AI Accelerators." In IEEE International Conference on Big Data (IEEE BigData 2022) December 17-20, 2022. Osaka, Japan.
3) S. Das, A. Dutta, S. Purohit, E. Serra, M. Halappanavar and A. Pothen, "Towards Automatic Mapping of Vulnerabilities to Attack Patterns using Large Language Models," 2022 IEEE International Symposium on Technologies for Homeland Security (HST), Boston, MA, USA, 2022, pp. 1-7, doi: 10.1109/HST56032.2022.10025459. **[Best Paper Award in Cyber Security Track]**
4) K. Panchal, S. S. Das, L. De La Torre, J. Miller, R. Rallo and M. Halappanavar, "Efficient Clustering of Software Vulnerabilities using Self Organizing Map (SOM)," 2022 IEEE International Symposium on Technologies for Homeland Security (HST), Boston, MA, USA, 2022, pp. 1-7, doi: 10.1109/HST56032.2022.10025443.
5) U Bhatia, S Chatterjee, A Ganguly, M Halappanavar, R Tipireddy, and R Brigantic. "Aviation Transportation, Cyber Threats, and Network-of-Networks: Conceptual Framing and Modeling Perspectives for Translating Theory to Practice." Accepted for publication in proceedings of the IEEE International Symposium on Technologies for Homeland Security (HST).
6) "FlipNet: Modeling Covert and Persistent Attacks on Networked Resources," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 2444-2451.
7) TH Bhuiyan, A Nandi, H Medal, and M Halappanavar. "Minimizing Expected Maximum Risk from Cyber-Attacks with Probabilistic Attack Success." In proceedings of the IEEE International Conference on Technologies for Homeland Security. May 10 - 12, 2016. Waltham, Massachusetts, USA.
8) "Identifying Vulnerabilities and Hardening Attack Graphs for Networked Systems." In proceedings of the IEEE International Conference on Technologies for Homeland Security. May 10 - 12, 2016. Waltham, Massachusetts, USA.
9) P Ramuhalli, M Halappanavar, J Coble, M Dixit. "Towards A Theory of Autonomous Reconstitution of Compromised Cyber-Systems." In proceedings of the 13th Annual IEEE Conference on Technologies for Homeland Security, Waltham, MA, 12 -- 14 Nov, 2013. **[Best Paper Award in Cyber Security Track]**

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354

1-888-375-PNNL (7665)

www.pnnl.gov