

Chapter 1

Generative AI for Advanced Cyber Defense

Moqsadur Rahman,¹ Aaron Sanchez,² Krish Piryani,³ Siddhartha Das,⁴ Sai Munikoti,⁵ Luis de la Torre Quintana,² Monowar Hasan,² Joseph Aguayo,⁵ Monika Akbar,¹ Shahriar Hossain,¹ and Mahantesh Halappanavar^{5*}

¹, *University of Texas at El Paso, El Paso, TX, USA*

², *Washington State University, Pullman, WA, USA*

³, *North Carolina State University, Raleigh, NC, USA*

⁴, *Purdue University, West Lafayette, IN, USA*

⁵, *Pacific Northwest National Laboratory, Richland, WA, USA*

*Corresponding Author: Mahantesh Halappanavar, hala@pnl.gov

Abstract: Due to an increasing pace of cyberattacks, there is a growing pressure to address vulnerabilities and detect ongoing cyberattacks in a timely manner. However, defenders need to traverse through siloed datasets for vulnerabilities, weaknesses, attack patterns, attack mechanisms, and mitigation actions to gain insights and make appropriate decisions. To address this challenging problem, we propose and develop a comprehensive Knowledge Graph (KG) and Retrieval-Augmented Generation (RAG) tool to enable cyber defenders to retrieve accurate information in an easy and interactive manner. We employ AI based techniques to accurately map MITRE's Common Vulnerabilities and Exposures (CVE) to Common Weakness Enumeration (CWE), and CWEs to Common Attack

Pattern Enumeration (CAPEC) and Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK). These mappings are then used to build a comprehensive knowledge graph that represents the information in a unified manner. The knowledge graph is used to retrieve specific information and to enable RAG to provide generative abilities with supporting evidence, and thus overcoming the traditional hallucination problems, as well as access to latest information. We expect our tool to provide significant support for robust cyberdefense.

Keywords: Generative AI, Retrieval augmented generation, Knowledge graphs, Cyber security

1.1. Introduction

The increasing sophistication and frequency of cyberattacks pose a critical challenge for cybersecurity professionals (illustrated in Figure 1.1). Defenders must always be ahead of adversaries that exploit known vulnerabilities, leverage advanced attack patterns, and develop novel tactics (§1.2). However, effective defense requires the integration of disparate data sources, such as information on vulnerabilities, weaknesses, attack patterns, and threat actor techniques (§1.3). Navigating these diverse sources to extract actionable insights is time-consuming and complex, blocking timely decision-making, and increasing the risk of successful attacks.

To address this challenge, this chapter introduces an advanced framework that combines Knowledge Graph (KG) technology (§1.5) with Retrieval Augmented Generation (RAG; §1.4) to enhance the analysis and understanding of cybersecurity concepts retrieved from multiple sources, such as vulnerability databases, attack patterns, weakness descriptions, threat intelligence, and

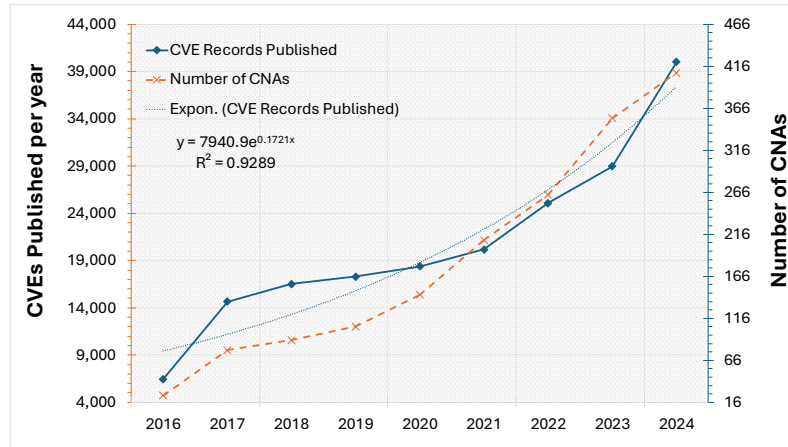


Figure 1.1: Growth of vulnerability disclosures from 2016 to 2024 through the MITRE CVE framework. Data is sourced from the [National Vulnerability Database \(NVD\) Dashboard](#).

response guidelines. Our approach does not aim to directly detect computer viruses or identify vulnerabilities. Instead, it provides a holistic perspective by connecting and contextualizing critical elements within cybersecurity knowledge bases. This perspective empowers cyber defenders to explore the relationships between vulnerabilities, weaknesses, attack patterns, and threat actor techniques thus facilitating better situational awareness and more informed decision-making.

Our system takes this a step further by integrating the knowledge graph with an RAG model. Unlike traditional generative AI models, which are often criticized for producing unverifiable information, our approach focuses on building meaningful connections between vulnerabilities, known weaknesses, and abstract mitigation strategies. By grounding the RAG’s responses in the knowledge graph and leveraging underlying embeddings, our system provides clear, evidence-based connections among cybersecurity concepts. From a generative AI (GenAI) perspective, this approach demonstrates how structured

knowledge integration can produce coherent, contextualized, and evidence-supported responses. This combination of GenAI and knowledge graph representation enables a reliable and insightful system for cybersecurity analysis.

This chapter also introduces evaluation techniques for assessing the effectiveness of the developed systems. By leveraging mappings between CVEs and CWEs (§1.3.1), we examine how well the proposed system can support reasoning and contextual understanding. We incorporate responses generated by large language models, such as LLaMA and ChatGPT, to benchmark our system against existing solutions. This evaluation offers insight into how knowledge graph-enhanced RAG systems may provide different perspectives compared to traditional approaches in terms of relevance and interpretability. To demonstrate the utility of our approach, we developed and evaluated variants of the RAG model: *(i)* a large language model without retrieval augmentation (§1.6.3), *(ii)* retrieval augmented generation (RAG), and *(iii)* RAG enhanced with the cyber knowledge graph (§1.6.2). Comparative analysis and case studies demonstrate the impact of the knowledge graph on improving the quality and relevance of responses. Our empirical results strongly support the value of integrating structured knowledge into the retrieval and generation processes (§1.6).

The key contributions of this chapter are as follows:

- Introduce a comprehensive knowledge graph that unifies and contextualizes key cybersecurity datasets, including CVE, CWE, CAPEC, and ATT&CK, into a single, interconnected framework.
- Demonstrate how a RAG model can be implemented with and without using a knowledge graph to reveal different perspectives for analysis.
- Discussion of experimental results and case studies highlighting the effec-

tiveness of the proposed approach, with comparisons to existing methods to highlight the effectiveness of responses.

1.2. Motivation and Related Work

Vulnerability management is a fundamentally hard economics problem. Starting with a fixed labor budget, engineers are charged with continuously maximizing the organization’s security posture over time. This is a complicated, high-dimensional task because of ongoing vulnerability research in the broader cybersecurity community, software lifecycle issues, latencies in detection, and remediation of vulnerabilities on organizational networks.

Furthermore, risk prioritization can change dramatically as new information becomes available. To manage this complexity, quantitative risk rankings are typically calculated based on various severity and exploitability scores. By regularly updating and reviewing each of these data points, it is possible to triage remediation work for a particular network on a particular day. But, there is one major drawback to these industry-standard risk rankings – they are not directly aligned with the way threat actors would compromise a target network. Skilled attackers with subject matter expertise in a given technology may be equally able to exploit a vulnerability ranked ‘9’ versus a vulnerability ranked ‘8’. Depending on the strategic value of the target, the adversary may also be willing to invest more cycles and resources into a given attack. Cybersecurity teams must be extremely careful to avoid mistaking precision for accuracy when quantifying risk.

Ideally, cybersecurity practitioners would have the option to start with the latest intelligence about specialized tactics, techniques, and procedures used by

attackers in real-world scenarios. This threat-based approach can be combined with vulnerability scans to better understand the total risk profile of a specific network. Historically, this analysis has not been done in a systematic way due to mathematical complexity. There are potentially large linkages to consider, including 300+ potential steps in the attack chain, 900+ types of weaknesses to target, and 303,423+ unique and documented vulnerabilities as of July, 2025 (§1.3.1).

This chapter proposes AI-based techniques to build a comprehensive knowledge graph to manage the relationships between individual vulnerabilities, weaknesses, attack patterns, attack mechanisms, and mitigation actions. This knowledge graph can be rapidly updated with new data from each of these separate data repositories and leveraged by Retrieval-Augmented Generation to provide reviewable citations for its conclusions.

AI-Supported Vulnerability Management. Vulnerability Managers scan digital assets with commercial and open-source tools to identify specific instances of vulnerabilities in their environment. These vulnerabilities are binned into Service Level Agreements (SLAs) using a risk-based triage approach, such as 30 days to remediate critical vulnerabilities, 60 days for high-risk vulnerabilities, and 90 days for moderate-risk vulnerabilities. The underlying algorithms vary widely for different organizations, but typically include some combination of: (1) Severity of the vulnerability (often expressed as the CVSS score¹ or a vendor-provided equivalent); (2) Exploitability of the vulnerability (often expressed as the EPSS score² or the CISA Known Exploited Vulnerabilities list³);

¹The Common Vulnerability Scoring System (CVSS)

²The Exploit Prediction Scoring System (EPSS)

³Known Exploited Vulnerabilities Catalog (KEV)

(3) Whether the vulnerable asset is publicly available and/or internet-facing; (4) The vulnerable asset’s defined role within the IT environment (Production, Disaster Recovery, Development, Test, etc.); (5) Value at risk of each business application, with a defined parent-child relationship to its underlying infrastructure and data; and (6) Any associated regulatory, compliance, and privacy concerns.

This traditional approach to vulnerability management is focused on documenting the attack surface. But, it does not factor in threat modeling and adversary emulation – which attackers may be targeting a given network, what are the strategic objectives of the adversary, and which tactics, techniques, and procedures would be utilized. Although some sources of threat intelligence helpfully enumerate specific lists of vulnerabilities, this can be error-prone, inconsistently available, and/or may require commercial subscription services.

An automation layer is needed to rapidly synthesize incoming threat information, linking it directly to individual vulnerabilities found on specific assets in the IT environment. The V2W-BERT framework [Das et al. \[2021\]](#) analyzes textual descriptions with a Transformer-based architecture to automate the mapping of CVEs to CWEs. It employs natural language processing, link prediction, and transfer learning to enhance classification accuracy and scalability. Achieving up to 97% prediction accuracy, V2W-BERT streamlines the traditionally labor-intensive manual process of mapping disparate data points, improving the scalability and efficiency of analyzing vast amounts of cybersecurity information. Other works have explored fine-tuning language models specifically for cybersecurity tasks, focusing on adapting general-purpose models like BERT to understand the unique terminology, syntax, and semantic relationships in cybersecurity-related texts [Aghaei et al. \[2022\]](#).

Once the text-based linkages are established in the dataset, this information is loaded into a knowledge graph, which is a structured representation of information that captures entities, their attributes, and their relationships in a graph format. It is designed to represent real-world data in a way that enables humans and machines to derive meaningful insights through querying, reasoning, and analysis. Knowledge graphs are used in scenarios where data is complex, interconnected, and needs to be contextualized for better understanding and actionable insights. Knowledge graphs are pivotal in representing, analyzing, and reasoning about complex relationships in cybersecurity and cyberdefense [Zhao et al. \[2024b\]](#), [Sikos \[2023\]](#), [Shen et al. \[2020\]](#), [Jia et al. \[2018\]](#), [Piplai et al. \[2020\]](#), [Liu et al. \[2022a,b\]](#).

Retrieval Augmented Generation (RAG) combines retrieval-based methods with generative models to enhance text generation by incorporating external knowledge [Lewis et al. \[2020\]](#). When used with knowledge graphs, RAG queries relevant graph data to ground the generative model’s output in structured, contextually accurate information [Sanmartin \[2024\]](#). This integration improves tasks like question-answering [Xu et al. \[2024\]](#), prompt-generation [Soman et al. \[2023\]](#), and knowledge-based reasoning [Hussien et al. \[2025\]](#). RAG-based approaches are increasingly emerging in generative AI for cybersecurity, enabling models to retrieve and integrate domain-specific knowledge for tasks like threat intelligence and vulnerability analysis [Rahman et al. \[2024\]](#), [Zhao et al. \[2024a\]](#).

By combining a Knowledge Graph of cybersecurity data with a Retrieval Augmented Generation tool, two key limitations of generative AI in the cybersecurity field are addressed: (*i*) AI chatbot have been demonstrated to generate erroneous, fictional, or unsupported information in response to user queries [Sun et al. \[2024\]](#), [Kumar et al. \[2023\]](#). However, cybersecurity is often safety-critical

with high consequences and thus a low tolerance for generative AI hallucination. Error-prone or misleading cybersecurity answers could potentially result in significant financial costs, business disruption, environmental damage, legal liabilities, and loss of life. *(ii)* The cybersecurity landscape is rapidly evolving (Table??). During 2024 alone, over 29,000 new vulnerabilities were formally documented by MITRE. Accurately and rapidly associating new vulnerabilities with weaknesses and attack patterns is vitally important to keep pace with the rate of change.

Practitioners can start with a particular cybersecurity concern (such as ransomware or lateral movement) and a finite labor budget for remediation activities. Our tool can be used to rapidly assess which remediation activities provide the largest Return on Investment (RoI) to address a specific threat. Another practical benefit of the proposed approach is related to operational efficiency. Remediation is often carried out by different stakeholders within a given IT organization. By grouping vulnerabilities according to specific weaknesses, it becomes more streamlined to divide up the workload across IT teams such as Identity and Access Management, Endpoint Management, Data Protection, Network Engineering, etc. Remediation progress can often be accelerated by tasking out the work in parallel based on functional IT responsibilities. We present the foundations for building a knowledge base for cyberdefense in the next section.

1.3. Foundations for Cyber Defense

In this work, we limit our study to the information derived from the following information catalogs: vulnerabilities, weaknesses, attack patterns, and adver-

Year	Qtr4	Qtr3	Qtr2	Qtr1	TOTAL
2024	11,073	8,591	11,716	8,697	40,077
2023	7,876	6,936	7,134	7,015	28,961
2022	6,231	6,448	6,365	6,015	25,059
2021	5,200	5,541	5,005	4,415	20,161
2020	4,387	4,170	5,011	4,807	18,375
2019	4,822	5,150	4,091	3,245	17,308
2018	3,614	4,350	4,613	3,935	16,512
2017	3,570	4,037	3,612	3,426	14,645
2016	1,590	1,713	1,775	1,379	6,457
2015	1,652	1,747	1,443	1,652	6,494

Table 1.1:

Summary of published CVE documents as reported by the [MITRE CVE Program](#).

Table: CVEs

sarial tactics, which are essential to build a comprehensive knowledge graph for cybersecurity. Key among these are 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) framework. These entities can be briefly described as follows (further details are provided in §1.3.1):

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, hardware, firmware, among others. that can lead to security vulnerabilities. It serves as a reference for identifying poor coding and development practices that might expose the system to potential attacks. By understanding these weaknesses, developers can avoid them during the coding process, ultimately creating more secure systems and 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 throughout the life-cycle of an attack. It serves as a knowledge base, outlining the steps attackers take during different phases of an attack, such as, reconnaissance, exploitation, and lateral movement. This information helps defenders identify and counteract malicious activities before they can cause significant damage.

1.3.1. Mapping Vulnerabilities, Weaknesses and Attack Patterns Using LLMs

A *weakness* is a flaw in the architecture, design, or implementation of cyber products such as software, operating systems, or hardware, which results in unintended and exploitable behavior. Common Weakness Enumerations (CWE) provide a comprehensive and hierarchical dictionary of these weaknesses. In comparison, a *vulnerability* consists of one or more weaknesses within a specific cyber product that can be exploited for malicious activities. Common Vulnerabilities and Exposures (CVE) reports document these vulnerabilities as identified by cybersecurity experts across the world in a timely manner. Understanding and addressing specific vulnerabilities in software products and protocols can be achieved by mapping them to hierarchically structured dictionaries that can provide details on attack mechanisms and potential mitigation actions. However, this mapping process is predominantly manual, making it slow and cumbersome as it cannot keep up with the vast number of new vulnerabilities discovered daily. As of July 2024, NVD lists 26,796 Critical, 69,093 High, 71,264 Medium, and 2,906 Low Vulnerabilities using CVSS-V3 scores⁴. Thus, automating the mapping of vulnerabilities to weaknesses is essential for timely mitigation. However, automation poses challenges such as semantic gaps between CVE and CWE languages, non-disjoint hierarchies in CWE classes (providing multiple paths to the same lower-level weaknesses), and limited training data for rare CWE classes with few or no mapped CVEs.

Das et al. introduced an innovative Transformer-based framework, V2W-BERT, which outperformed previous methods in mapping CVEs to the CWE

⁴[National Vulnerability Database \(NVD\) Dashboard](#)

hierarchy at finer levels of granularity, proving particularly effective for rare CVEs [Das et al. \[2021\]](#). V2W-BERT employed a Siamese network of Bidirectional Encoder Representations from Transformers (BERT) models and formulated the task of CVE to CWE mapping as a link prediction problem. A BERT model is designed to pre-train deep bidirectional representations from unlabeled text by considering the context on both sides of a text token simultaneously during training. BERT is trained on a large text corpus, developing a deep understanding of language, which is beneficial for various language processing tasks. Pre-trained BERT models can be enhanced with custom layers to adapt to a wide range of Natural Language Processing (NLP) tasks. V2W-BERT harnesses this capability to transfer domain knowledge for CVE mapping. The Siamese model to embed semantically different text forms in CVEs and CWEs into a common space, enabling mapping through link prediction. An illustration of the V2W-BERT is provided in [Figure 1.2](#). The Link Prediction (LP) component is designed to establish connections between a CVE and the available CWEs, even in scenarios with limited or non-existent training instances. To maintain contextual integrity during LP updates, we incorporate a Reconstruction Decoder (RD) component. When the BERT encoder processes a CVE/CWE description, the resulting last hidden state is fed into the Masked Language Model (LM) and optimized for Masked tokens. The Cross-Entropy loss function is used to minimize the discrepancy between the original input and the reconstructed tokens. Both LP and RD utilize the hidden states generated by BERT, and their trainable layers are concurrently updated by considering both link classification loss and reconstruction loss. This approach allows V2W-BERT to train for link classification while preserving context. Using vulnerability and weakness reports from MITRE and the National Vul-

nerability Database⁵, V2W-BERT achieved up to 97% prediction accuracy for randomly partitioned data and up to 94% prediction accuracy in temporally partitioned data and thus provided the foundation for the framework proposed in this chapter.

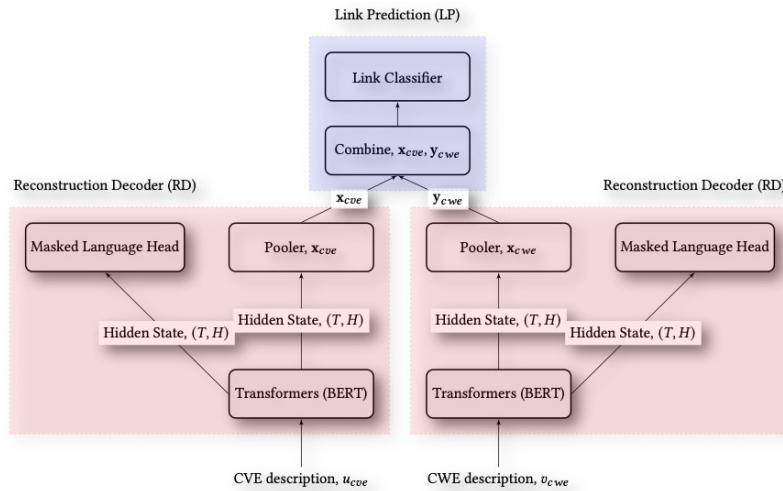


Figure 1.2: An architectural overview of the V2W-BERT framework for mapping CVEs to CWEs formulated as a hierarchical multi-class classification problem.

In VWC-MAP, Das et al. extend V2W-BERT by efficiently mapping CWEs to Common Attack Pattern Enumeration and Classifications (CAPEC) using a Text-to-Text model (Google T5) for learning the relationships, as well as a link prediction approach by incorporating the hierarchical relationships of CAPECs through positive and negative samples Das et al. [2022a,b]. We employ both the frameworks to build the cyber knowledge graph presented in §1.5. In the next section, we provide a brief introduction to retrieval augmented generation as a primary vehicle for information retrieval and interaction.

⁵<https://nvd.nist.gov/>

1.4. Retrieval Augmented Generation

Large language models (LLM) have become a cornerstone in the field of natural language processing (NLP), fundamentally transforming how we approach various tasks such as text generation, summarization, and information retrieval. Language models are trained using large corpora of text data to develop an understanding of linguistic patterns, semantics, and context, enabling them to perform a variety of tasks. In traditional information retrieval, language models are employed mainly to match user queries with relevant documents in a corpus, often by computing relevance scores and consequently ranking documents. Here, retrieval relies on basic techniques such as keyword matching or vector space models, where documents and queries are represented as vectors (e.g., TF-IDF) in high-dimensional space. These methods predominantly consider surface-level text features, which limits their ability to grasp the deeper semantic relationships between terms. With the advent of LLMs, such techniques are augmented by embedding-based retrieval systems. Specifically, Encoder models such as BERT [Kenton and Toutanova \[2019\]](#) or BGE [Chen et al. \[2024a\]](#) embed documents and queries into dense vectors that capture semantic meaning, thereby improving the precision of retrieval tasks by enabling better matching of query intents with document content.

Retrieval-Augmented Generation (RAG) further advances the field of information retrieval by combining the strengths of embedding-based retrieval systems with the generative capabilities of language models [Lewis et al. \[2020\]](#). In RAG, an external knowledge base, such as a document corpus or a database, is integrated with the language model. When a query is received, RAG first retrieves relevant documents (passages) from this external knowledge base and then uses a generative model to synthesize a coherent response, utilizing the

retrieved information. RAG systems are particularly useful for complex queries that cannot be sufficiently answered only by the information the model was originally trained on. By accessing an external corpus, RAG systems enrich their outputs with contextually relevant information, leading to better responses, as well as evidence for their responses.

Despite significant improvements, standard text-only RAG systems have several shortcomings [Fan et al. \[2024\]](#), [Chen et al. \[2024b\]](#). The retriever relies heavily on textual similarity and may fail to capture the latent, complex relationships present in the datasets that require multi-hop reasoning across datasets or external sources like ontologies or knowledge graphs. As a result, text-only RAG can lead to lower retrieval accuracy and often results in irrelevant documents being retrieved from the corpus, thereby limiting the effectiveness of the generative model [Procko and Ochoa \[2024\]](#).

RAG with structured information in the form of knowledge graphs can overcome some of these limitations found in text-only RAG systems [Han et al. \[2024\]](#). Such systems enhance retrieval capabilities by representing knowledge database in a graph format, where nodes represent entities or concepts, and edges represent the relationships between them. This structured representation can capture complex contextual and relational information, which is often overlooked by linear text formats. In graph-based RAG system, retriever traverses via the graph database and can unravel underlying connections between disparate pieces of information, allowing for a richer, more adequate retrieval output. Furthermore, graph-based methods facilitate efficient query processing through advanced indexing and searching mechanisms specifically tailored to exploit the graph structure. We discuss constructing the cyber knowledge graph in the next section.

1.5. Knowledge Graph & Querying

A graph $G = (V, E)$ is a mathematical abstraction consisting of a vertex set V representing unique entities and an edge set E representing pairwise relationships on vertices. A knowledge graph extends the scope of a graph by adding: (i) Labels: assigned to nodes and edges to categorize them into different types (classes or predicates), and (ii) Attributes (Properties): additional information or characteristics associated with nodes and edges, usually stored as key-value pairs [Ji et al. \[2022\]](#). Knowledge graph (KG) adds semantic richness, meaning through ontology, with the flexibility of a graph to represent complex relationships without a rigid database schema. KG also adds heterogeneity to the vertices and edges by using labels/attributes. Two key technologies enabling KG are: (i) Resource Description Framework (RDF)⁶: World Wide Web Consortium (W3C) standard model for representing information in a graph form that enables data interchange and facilitates the sharing and reuse of structured information across different applications; and (ii) SPARQL Protocol and RDF Query Language (SPARQL)⁷: W3C standard query language and protocol for querying and manipulating RDF graph data stored in an RDF store on the Web. SPARQL queries are written in a declarative manner and specify what data to retrieve by using graph pattern matching to find data.

A knowledge graph, such as the one built using Neo4j, offers an intuitive way to model relationships between different entities in the cybersecurity domain [Vukotic et al. \[2015\]](#). Entities such as vulnerabilities, weaknesses, attack patterns, and adversarial tactics (details in §1.3.1) can be represented as nodes, with the relationships between them as edges. This interconnected structure

⁶<https://www.w3.org/RDF/>

⁷<https://www.w3.org/TR/rdf-sparql-query/>

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 the current flat and siloed databases.

1.5.1. Graph Schema

The knowledge graph schema in Neo4j consists of various types of nodes and relationships that represent the underlying data in a structured and interconnected manner. In this schema, entities such as vulnerabilities, weaknesses, attack patterns, and adversarial tactics are captured as nodes, each having specific properties that describe their characteristics. The relationships between these nodes illustrate the connections and interactions among these entities, providing a comprehensive view of the cyber security domain.

Nodes. Different node types in the cyber knowledge graph are as follows:

CVE: CVE nodes represent known vulnerabilities within the knowledge graph (Figure 1.3(a)). These nodes contain key properties that describe the nature, severity, and impact of the vulnerability. As of 24 December 2024, there are 274,646 CVEs listed in the NIST National Vulnerability Database. However, our dataset consists a total of 245,125 CVEs collected as of March, 2024 [Das et al. \[2021\]](#). The properties of CVE nodes are: $\langle \text{id, name, description, base_score, category, exploitability_score, impact_score, related_cwe_ids, severity, Embedding} \rangle$.

CWE: CWE nodes represent common weaknesses in the knowledge graph.

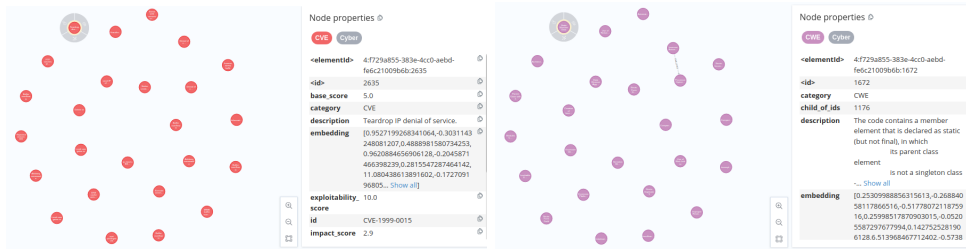


Figure 1.3: (a) CVE Node (b) CWE Node

These nodes contain properties that describe the nature and relationships of these weaknesses. In total, there are 963 CWE nodes. The properties of CWE nodes are: \langle category, child_of.ids, description, embedding, id, name \rangle . Figure 1.3(b) shows an example of a CWE node.

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 (Figure 1.4(a)). There are 615 CAPEC nodes. The properties of CAPEC nodes are: \langle category, childOf, description, embedding, id, likelihood, mitigations, name, severity \rangle .

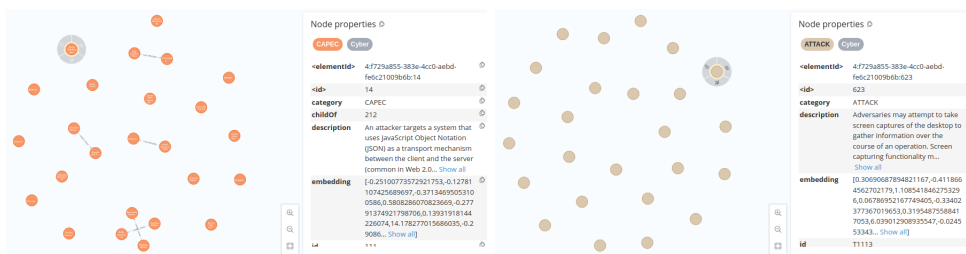


Figure 1.4: (a) CAPEC Node (b) ATT&CK Node

ATT&CK: The ATT&CK node represents a critical element that encapsulates detailed information about specific cyber threats or malicious activities (Figure 1.4(b)). The structure of the Attack node is built upon a set of well-defined properties, which provide a comprehensive understanding of the at-

tack’s characteristics and context. There are 1043 ATT&CK nodes with the following properties: $\langle \text{id, name, description, type, category, embedding} \rangle$.

Relationships. The total number of relationships in the graph is 270,093. Different types of relationships are as follows:

CVE_CWE: links a CVE with a CWE and helps identify the weakness categories (CWEs) that a CVE belongs to. These relations help in understanding the root causes of vulnerabilities (Figure 1.5(a)).

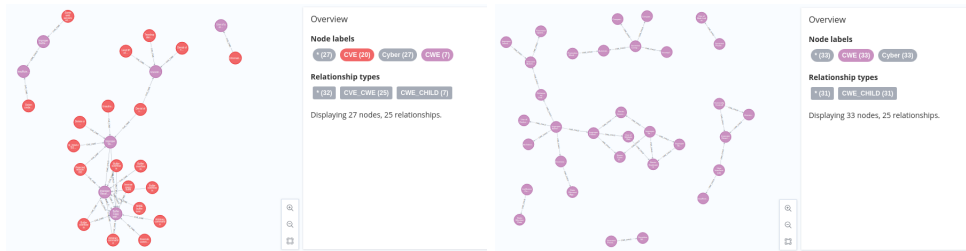


Figure 1.5: (a) CVE-CWE Relationship (b) CWE-CHILD Relationship

CWE_CHILD: reflects a hierarchical structure within the CWE database. A specific CWE (child) is linked to a more general category (parent) (Figure 1.5(b)). These edges enable a structured and organized analysis of weaknesses.

CWE_CAPEC: indicates a connection between a CWE and a CAPEC entry, showing a specific weakness (CWE) can lead to an attack pattern (CAPECs) (Figure 1.6(a)). These relations assist in visualizing which weaknesses are targeted by specific attack strategies.

CAPEC_CHILD: denotes a hierarchical relationship between CAPEC entries, where one CAPEC attack pattern is a more specific subtype (child) of another, more general CAPEC pattern (parent) (Figure 1.6(b)). It reflects the structured nature of attack patterns.

CAN_PRECEDE: denotes an ordering or a sequence in attack patterns (Fig-

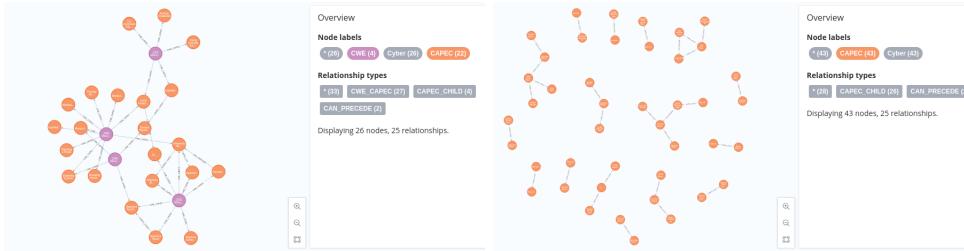


Figure 1.6: (a) CWE-CAPEC Relationship (b) CAPEC-CHILD Relationship

ure 1.7(a)). In a cybersecurity context, one attack method may precede another, implying that the occurrence of one tactic increases the likelihood of the next step in the attack chain or a prerequisite for a given tactic.

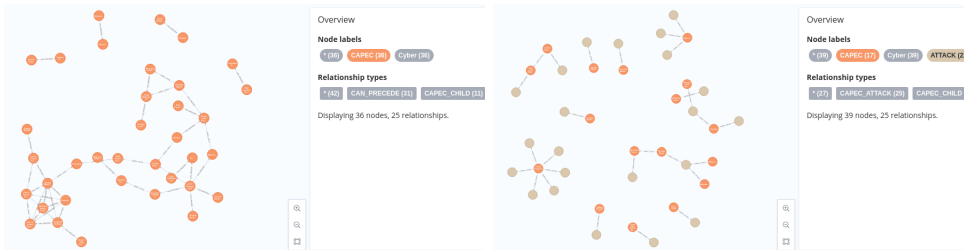


Figure 1.7: (a) CAN-PRECEDE Relationship (b) CAPEC-ATT&CK Relationship

CAPEC_ATTACK: connects a CAPEC entry to a specific attack (Figure 1.7(b)). A CAPEC entity represents the method used in an attack, thereby categorizing the attack under a broader pattern of known threats.

1.5.2. Neo4j Knowledge Graph Implementation

Implementing a Neo4j Knowledge Graph for cybersecurity involves several essential steps to effectively manage and analyze interconnected data. The key stages/steps that we followed in our work are:

Data Collection: The first step in the Neo4j Knowledge Graph implemen-

tation 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. Preprocessing aims to make the raw data usable by converting it into a clean, structured format optimized for efficient querying and relationship mapping in 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.

Importing Data into Neo4j: Once the entities, properties, and relationships are defined, the data is ready to be imported into Neo4j. The preprocessed data is loaded into the Neo4j database using Cypher queries or batch import methods, 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.

1.5.3. Cypher Queries

Cypher is Neo4j's query language, which provides an intuitive and powerful means to retrieve information from a graph database. Whether performing basic queries or tackling more advanced data retrieval tasks, Cypher excels in handling complex relationships and patterns in large datasets. It is particularly valuable in domains like cybersecurity, where understanding interconnected data is crucial. Below are some examples of how Cypher can be used:

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 count nodes representing vulnerabilities, weaknesses, attack patterns, or attacks. This is essential for understanding the distribution and prevalence of different entities in the dataset.

Example:

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

Response:

```
242504
```

Counting 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.

Example:

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

Response:

```
6931
```

Node Information Extraction. Cypher allows for extracting detailed information about specific nodes in a Neo4j database. For example, in a cybersecurity knowledge graph, retrieving properties of a node representing a vulnerability can provide insights into its characteristics and associated metadata. This is useful for analyzing vulnerabilities, their relationships, and their context within the dataset.

Example:

```
MATCH (n:CVE {id: 'CVE-2021-34527'})
RETURN n;
```

Response:

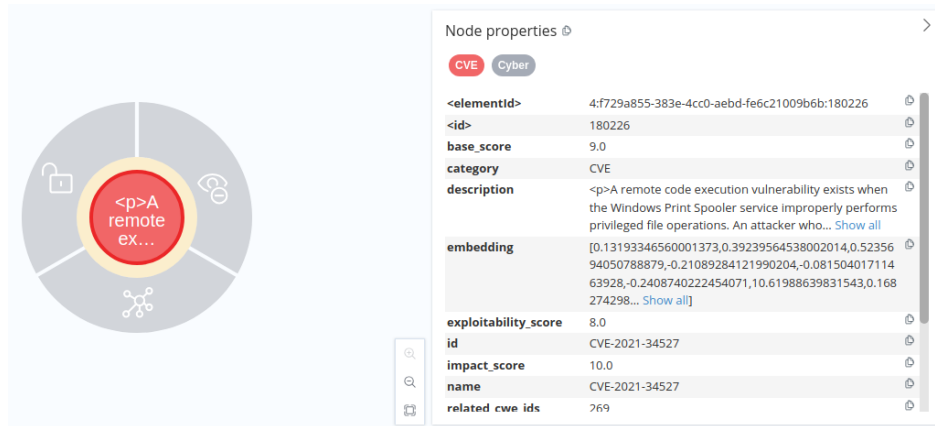


Figure 1.8: Extracting Node Information.

Exploring Cybersecurity Relationships. In 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 (ATT&CK), providing a comprehensive view of how these elements are related. Cypher 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 can build more robust defense strategies, focusing resources on high-risk areas and understanding the full scope of potential cyber threats.

Example:

```
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:

Example:

```
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:

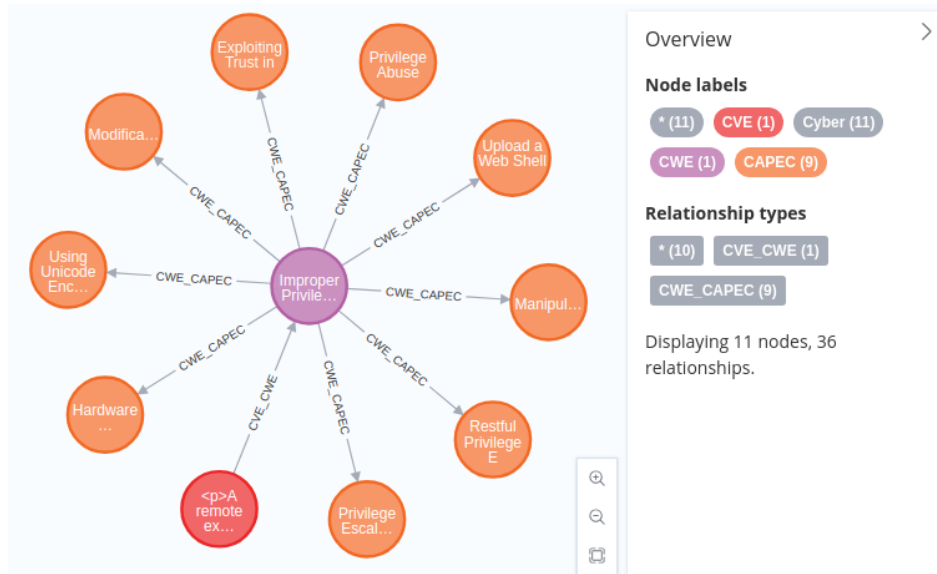


Figure 1.9: Finding CAPECs for a given CVE.

These examples demonstrate how Cypher queries can facilitate advanced data analysis and threat intelligence in Neo4j, making it an indispensable tool for managing and understanding interconnected data. In the next section, we discuss how the cyber knowledge graph can be coupled with retrieval augmented generation for interactive information retrieval.

1.6. Evaluation and Results

In this section, we provide an empirical evaluation of the proposed approaches. We begin with a general evaluation of the RAG-based tool using a broad set of example queries and demonstrate the effectiveness of using a knowledge graph-based RAG to generate responses (§1.6.1). We then provide specific test scenarios by using prediction of CWEs for a given CVE. We describe the experimental setup, metrics, and experimental results for GraphRAG (§1.6.2)

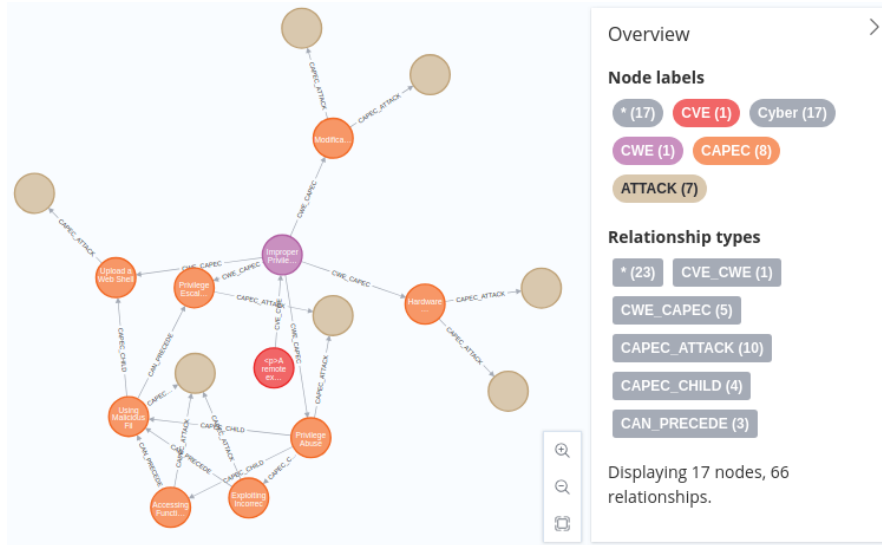


Figure 1.10: Finding ATT&CKs from a given CVE

and GPT4o (§1.6.3). By using a variety of modifications to the quality of input query, we demonstrate the robustness of each method for prediction.

1.6.1. RAG-Based Response Generation

Methodology. RAG utilizing Knowledge Graph 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 (ATT&CK). A brief breakdown of our RAG approach utilizing Knowledge Graph is as follows:

1. 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. For example, a user may ask: “*What are the possible attacks related to CWE 89?*” The system converts this prompt into a Cypher query as follows:

```
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
```

2. 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. The resulting 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.

3. Preprocessing the Cypher Query Output: After the data is retrieved from the knowledge graph, it needs to be preprocessed before being fed into an LLM as a context. This step involves cleaning the data, structuring it for easy interpretation, and ensuring that only the most relevant information is passed to the LLM. For example, a query may return related nodes, including CWE, CAPEC, and ATTACK nodes, along with embedding vectors that are not required for response generation. For our sample prompt, the response would be:

```
{
```

```
"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, .....]}
],
}
```

In post-processing, we focus on removing unnecessary data such as embedding vectors, leaving only the relevant node attributes like id, name, etc., resulting in the following output:

```
{
  "nodes": [
    {"id": "CWE-89", "name": "SQL Injection"},
    {"id": "CAPEC-123", "name": "Injection Attack"},
    {"id": "T1134", "name": "Privilege Escalation"}
  ],
}
```

4. 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.

Examples. We created a set of queries to test different aspects of cyber defense – from simple queries on the number of CVEs to complex queries linking CVEs to CAPECs. We use these queries to test the effectiveness of using a knowledge

graph-based RAG compared to a response obtained directly from an LLM. For our experiments, we have used the pretrained Llama3 model for all the queries in this subsection.

Example 1:

- **User query:** How many CVEs are there? Give me the approximate count.
- **Llama3 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.
- **GraphRAG Response:** The approximate count of CVEs in the graph database is 242,504.
- **Observation:** GraphRAG responded with precise information.

Example 2:

- **User query:** Tell me about CWE 152 and its respective MITRE CAPEC and ATT&CK techniques.
- **Llama3 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.

- **GraphRAG 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.

- **Observation:** GraphRAG was able to find CWE 152 and its related CAPEC and ATT&CK information more precisely and accurately by utilizing the

knowledge graph. In contrast, Llama3 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.
- **Llama3 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.

- **GraphRAG 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.

- **Observation:** GraphRAG was able to accurately retrieve related CAPECs for CVE-2021-34527, providing detailed and relevant information using the knowledge graph, while Llama3 incorrectly stated that no CAPECs were associated with this CVE. The GraphRAG 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.
- **Llama3 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. The exploitability score for CVE-1999-0009 is low, as specific conditions must be met for the vulnerability to be successfully exploited, such as using a particular network protocol and sending a large volume of packets to the affected system. Additionally, this vulnerability was first reported in 1999 and has since been patched by Microsoft, making it less likely to be successfully exploited on modern systems.

While CVE-1999-0009 may still pose a risk to older systems, its impact and exploitability are relatively low compared to more recent vulnerabilities. It is essential to keep systems up to date with the latest security patches to protect against vulnerabilities like this one.

- **GraphRAG Response:** The impact score of CVE-1999-0009 is 10.0, and the exploitability score is 10.0. This indicates that the vulnerability has a significant potential impact on the affected system and is highly exploitable. It is crucial to address this vulnerability as soon as possible to prevent potential attacks.
- **Observation:** The GraphRAG 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, Llama3's response was more descriptive but lacked specific scores and provided a broader explanation regarding the conditions and history of the vulnerability. The GraphRAG system demonstrated greater precision by offering the exact scores relevant to the query, while Llama3 provided a less focused and verbose response.

1.6.2. CWE Predictions Using RAG

Methodology. We used the following steps to predict the most appropriate CWEs for a CVE description: 1) **Compute Embeddings:** For each testing sample, the embedding of the respective CVE description is extracted using an embedding model named BAAI/bge-small-en-v1.5. The BAAI/bge-small-en-v1.5 model is an English embedding model designed for sentence similarity and retrieval tasks. It offers a compact architecture with 384 dimensions, optimized

for efficient performance while maintaining competitive accuracy. **2) Compare CVE Node Embeddings:** The computed embedding is compared against all CVE node embeddings stored in the knowledge graph database. **3) Top-3 CVE Selection:** Based on the similarity scores, the top 3 CVE nodes with the highest similarity to the testing sample are identified. **4) CWE Mapping:** For the selected top 3 CVE nodes, the associated CWE nodes are extracted using Cypher Queries through the CVE_CWE edges in the knowledge graph. **textbf5) Variation in Predicted CWE Nodes:** The number of CWE nodes associated with a CVE can vary due to multiple or zero CWE mappings. As a result, the number of predicted CWE nodes may differ for each sample, with 2 or more CWE nodes (at different levels of the CWE hierarchy) being associated in some cases.

Measuring Accuracy. The accuracy of the RAG tool is measured based on its ability to predict the correct CWE(s) for a given CVE description. We conduct the evaluation on a dataset of 250 CVE samples. Each sample consists of a CVE description along with its associated correct CWE(s), against which the predicted CWE list is compared. We compute the accuracy as follows.

For each CVE description, the tool predicts a list of CWE nodes based on the top 3 most similar CVE nodes in the knowledge graph. If any of the predicted CWEs matches the correct CWE for the given CVE description, it is considered a true prediction. Otherwise, it is classified as an incorrect prediction. Accuracy is defined as the percentage of samples for which at least one of the predicted CWEs matches the correct CWE. Mathematically,

$$\text{Accuracy} = \frac{\text{Number of True Predictions}}{\text{Total Number of Samples}} \times 100. \quad (1.1)$$

In this case, the number of true predictions corresponds to the samples where the correct CWE appears in the predicted CWE list.

The following example shows a sample prediction result for a given CVE description: “*Web sessions in the management interface in Palo Alto Networks PAN-OS software do not expire in certain situations, making it susceptible to unauthorized access.*” The correct (ground-truth) CWE for this description is CWE-384. The RAG based tool predicted the following CVEs as top three: {CVE-2024-0008, CVE-2020-2039, CVE-2017-7216}. Using the knowledge graph, the corresponding CWEs for these CVEs are: {CWE-384, CWE-400, CWE-200}. Since the correct CWE appears in the top-3 list, we consider this as a true prediction.

1.6.2.1. Robustness Against Text Removal

When provided a full and accurate description of a CVE, our Retrieval-Augmented Generation (RAG) tool was able to predict the correct CWE across the entire testset. Therefore, to evaluate the robustness and performance of the tool, we conducted experiments using CVE descriptions with systematic alterations. These alterations involved removing portions of text from the beginning, end, middle, and randomly, at varying percentages (10%, 20%, 30%, ..., 90%). The resulting impact on the accuracy of the RAG tool was assessed, with the accuracy trends illustrated through bar diagrams.

Accuracy Trends Under Different Text Removal Lengths. Understanding how the accuracy of a tool or system is affected by the removal of textual content is critical for evaluating its robustness and dependency on specific contextual information. This section investigates accuracy trends under vari-

ous text removal strategies, including removal from the beginning, end, middle, and random positions (Figure 1.11). These scenarios simulate real-world challenges, such as incomplete data, missing context, or unpredictable information loss. By analyzing the impact of different removal patterns, we aim to uncover the system’s sensitivity to the location and proportion of missing text, providing valuable insights into its resilience and adaptability.

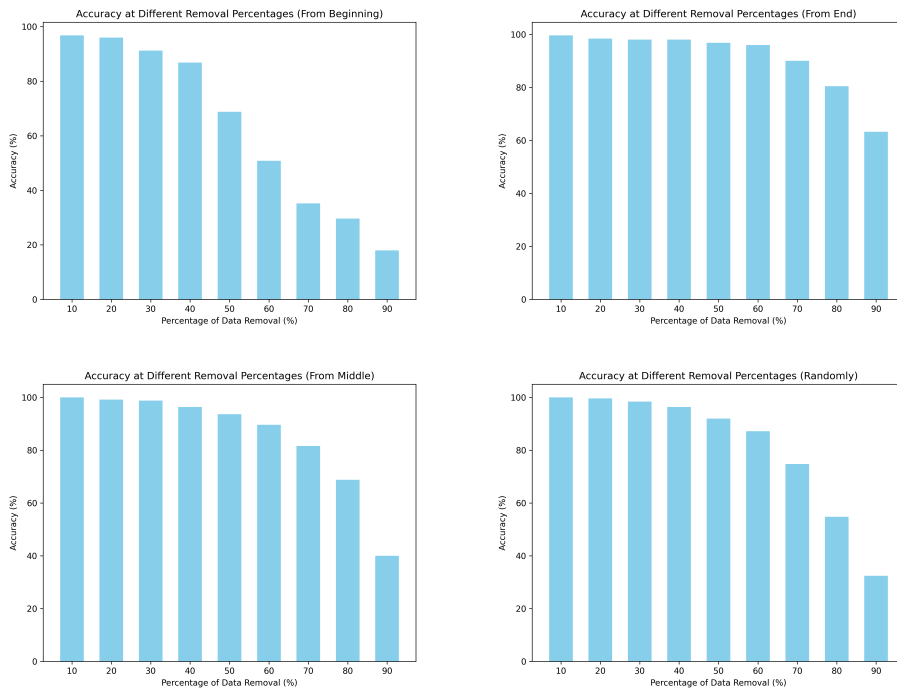


Figure 1.11: Accuracy of prediction at different text removal percentages.

- Removal from Beginning:** Text portions were progressively removed starting from the beginning, simulating scenarios where introductory context is absent. As the percentage of removed text increased, the accuracy exhibited a steady decline, emphasizing the tool’s dependence on contextual information at the start.

- **Removal from End:** Portions were removed from the end of the text, representing scenarios where conclusive information is unavailable. The accuracy drop was slower compared to removal from the beginning but still significant at higher removal percentages.
- **Removal from Middle:** Central portions of text were removed, mimicking cases where critical intermediate information is missing. Interestingly, the tool showed relatively better resilience compared to the other removal strategies for lower percentages of middle removal.
- **Random Removal:** Text was randomly removed at various percentages, representing unpredictable data loss. The accuracy exhibited a gradual decline, though it demonstrated better resilience than removal from the beginning, likely due to the distribution of retained information.

Observations and Insights.

- **Dependence on Context:** The results highlight the critical role of beginning and end portions in the RAG tool's performance, as accuracy declined sharply when these sections were removed.
- **Resilience to Middle and Random Removal:** The tool was moderately robust to middle and random removal at lower percentages, suggesting its ability to generalize when some contextual information remains intact.
- **Accuracy Trends:** The bar diagrams for each removal strategy underline that higher percentages of removal universally degrade accuracy, though the extent varies by the type of alteration.

This analysis provides a comprehensive understanding of the RAG tool's dependency on the availability and location of textual context. It also emphasizes the need for careful preprocessing and input handling to mitigate the

impact of incomplete or noisy data. These insights will inform future improvements to enhance the tool’s robustness and adaptability in real-world scenarios.

1.6.2.2. Robustness Against Text Rewriting

The experiment aimed to evaluate the performance of the RAG tool when provided with rewritten or summarized versions of original CVE descriptions. These rewritten texts were generated using GPT-4o API, ensuring varying lengths while maintaining the core meaning of the descriptions. The goal was to assess the tool’s robustness and accuracy under scenarios where the input descriptions are modified to different lengths.

Accuracy Trends Under Different Text Rewritten Lengths. Our approach for rewriting text is as follows. The CVE descriptions were rewritten or summarized using GPT-4o. The lengths of the rewritten texts were controlled by the following prompt: *“I want you to rewrite or summarize the following text with around {expected_lengths} words. The text is: {description}.”* The {expected_lengths} were determined based on the percentage of the original text length, ranging from 10% to 100%.

For each rewritten text length category (10%, 20%, ..., 100%), 250 samples were selected to generate a probabilistic result efficiently. These samples included original CVE descriptions and their rewritten versions, which were used as inputs for the RAG tool.

GPT-4o demonstrated excellent performance in generating rewritten texts that adhered closely to the specified lengths while preserving essential content. A sample of the rewritten outputs at different lengths for a given CVE description are provided below:

Original Description: “A cross-site scripting (XSS) vulnerability in Palo Alto Networks PAN-OS software enables a malicious authenticated read-write administrator to store a JavaScript payload using the web interface on Panorama appliances. This enables the impersonation of another authenticated administrator.”

Rewritten Texts at Various Lengths:

- **10% length:** “XSS vulnerability exploitation.”
- **20% length:** “XSS vulnerability in PAN-OS allows admin impersonation.”
- **30% length:** “XSS vulnerability in PAN-OS allows admin impersonation via JavaScript payload.”
- **40% length:** “An XSS vulnerability in PAN-OS allows malicious admins to impersonate others via JavaScript.”
- **50% length:** “An XSS vulnerability in PAN-OS allows malicious admins to store JavaScript, impersonating other authenticated admins.”

The prediction results for different percentages of rewriting are plotted in Figure 1.12. For each length category, 250 samples were used to provide a probabilistic analysis, yielding insights into the tool’s performance under varying input lengths.

Observations. Our experimental evaluation demonstrate the following: (i) *Length Sensitivity:* The accuracy of the RAG tool demonstrated a noticeable variation with the length of the input text. Shorter descriptions (e.g., 10%, 20%) tended to result in reduced accuracy due to the loss of critical contextual information. (ii) *Optimal Lengths:* Rewritten texts in the range of 50%-70% of the original length maintained a balance between brevity and content, yielding comparable accuracy to the original descriptions. (iii) *Performance at Full*

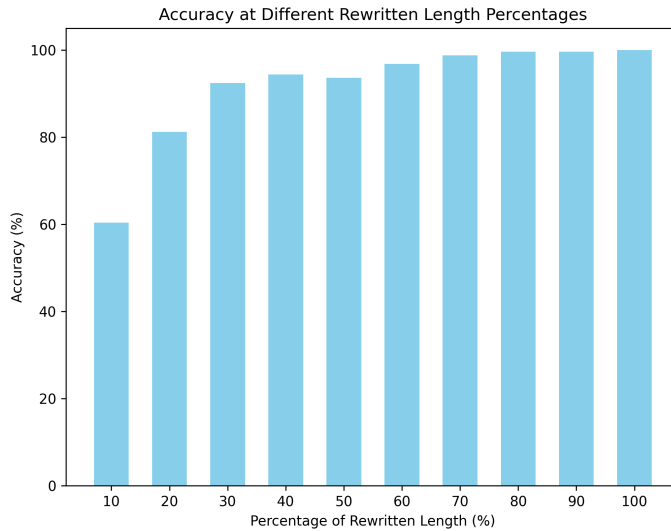


Figure 1.12: Accuracy of prediction at different text rewriting length restrictions.

Length (100%): The tool performed best when provided with complete descriptions, as expected, since all context and details were retained. Thus, our experiments highlight the importance of input length and content preservation when using RAG tools, paving the way for future enhancements to improve robustness across varying input scenarios.

Unseen CVEs. The performance of the current Retrieval-Augmented Generation (RAG) system was evaluated on recently published CVEs that have not yet been incorporated into the knowledge graph. The experiment was conducted to assess how effectively the system handles CVEs that are entirely unseen. A total of 250 samples, all unknown to the knowledge graph, were used for testing. The number of correct predictions was 191, with a resulting accuracy of 76.40

1.6.3. CWE Predictions Using GPT4o

Methodology. We use the following steps to predict the top CWEs for a given CVE description using GPT-4o:

1) **Prompt Design:** For each testing sample, a request was constructed to call the `/v1/chat/completions` API using the GPT-4o model. The system was configured with specific content to provide context for CWE prediction. The table below outlines the key-value structure of the system prompt:

Key	Value
model	gpt-4o
url	/v1/chat/completions
role	system
content	<i>“Tell me the quantity most likely CWE weakness descriptions that correspond to the vulnerability description.”</i>

2) **Token Limitation and Dataset Partitioning:** Due to the 90,000-token limit imposed by the Tier-1 batch configuration, the evaluation dataset was partitioned into separate files, each containing approximately 250 prompts. This ensured that the testing samples were processed without exceeding the token constraints. (3) **System Instructions and Output:** The GPT-4o model was instructed to generate predictions using two different prompt variations: *i*1) Single Prediction: Return the most relevant CWE, and *i*2) Top-3 Predictions: Return a list of the 3 most relevant CWEs.

Measuring Accuracy. Similar to the evaluation of RAG (§1.6.2), we conduct the GPT-4o experiments on a dataset of 250 CVE samples. Each sample included a CVE description along with its associated CWE(s) from the V2W-BERT mapping and the Baseline (NIST) CWE mapping. The predicted CWE list was compared against these mappings to assess accuracy using Equa-

tion 1.1.

For each CVE description, GPT-4o generated predictions based exclusively on its internal knowledge and the instructions provided in the system role content. The accuracy of these predictions was evaluated as follows. For each CVE description, GPT-4o predicts a list of CWE-IDs. We propose two evaluation criteria: (i) Single Prediction: The prediction was considered accurate only if the predicted CWE-ID perfectly matched the correct CWE-ID. (ii) Top-3 Predictions: The prediction was considered successful if any of the top 3 predicted CWE-IDs matched the correct CWE-ID.

As an illustrative example, we present a sample prediction result from GPT4o for the given CVE description: *“Tell me the 3 most likely CWE weakness description that corresponds to the vulnerability description and format the output as {cwe-Id,cwe-Id,cwe-Id}”* and user content: *“Web sessions in the management interface in Palo Alto Networks PAN-OS software do not expire in certain situations, making it susceptible to unauthorized access.”*. The predicted results are as follows: (i) Baseline (NIST) CWE: CWE-613, (ii) V2W-BERT mapped (CWE): CWE-384, and (iii) GPT-4o predicted CWE list {CWE-613, CWE-384, CWE-287}.

Prediction Results. The table below summarizes the accuracy results for both single and top-3 CWE predictions. Notably, validation against the data predicted by V2W-BERT yielded higher accuracy compared to validation against the Baseline (NIST) CWE.

Table: Prediction accuracy Baseline (NIST) and V2W-BERT.

Date	Prediction Type	Baseline (NIST) Accuracy	V2W-BERT Accuracy
3/23/2024	Single Prediction	60.75%	73.85%
	Top-3 Predictions	70.35%	84.15%

1.6.3.1. Robustness Against Text Rewriting

To evaluate performance on mapping CVE to CWE, rewritten (summarized) versions of original CVE descriptions were created using GPT-4o, which was instructed to vary the lengths of rewriting while ensuring that the core meaning remained intact. The goal was to assess the tool’s robustness and accuracy when the input descriptions were modified to different lengths.

The CVE descriptions were rewritten or summarized using GPT-4o. All the 250 samples were selected for each rewritten text length category: 10%, 20%, ..., 100%. The samples included original CVE descriptions and their rewritten versions, which were used as inputs to GPT-4o. The lengths of the rewritten texts were controlled using the following API batch text prompts:

System Content: “I want you to rewrite or summarize the vulnerability description in JSON format, with word counts at approximately 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100% of the original length.”

User Content: “Vulnerability description: \langle Vulnerability description \rangle ”

GPT-4o demonstrated excellent performance in generating rewritten texts that adhered closely to the specified lengths while preserving essential content. The results were plotted to visualize the impact of rewritten text lengths on GPT-4o accuracy. A sample of the rewritten outputs at different lengths for a CVE description is provided below:

Original Description (CVE-2024-1010): *“A vulnerability classified as problematic has been found in SourceCodester Employee Management System 1.0. This affects an unknown part of the file edit-profile.php. The manipulation of the argument fullname/phone/date of birth/address/date of appointment leads to cross site scripting. It is possible to initiate the attack remotely. The associated identifier of this vulnerability is VDB-252279.”*

Rewritten text:

20% length: “Vulnerability in SourceCodester leads to XSS. ID: VDB-252279.”

50% length: “SourceCodester Employee Management System 1.0 has a vulnerability in edit-profile.php that manipulates fullname/phone/other parameters, enabling remote XSS. ID: VDB-252279.”

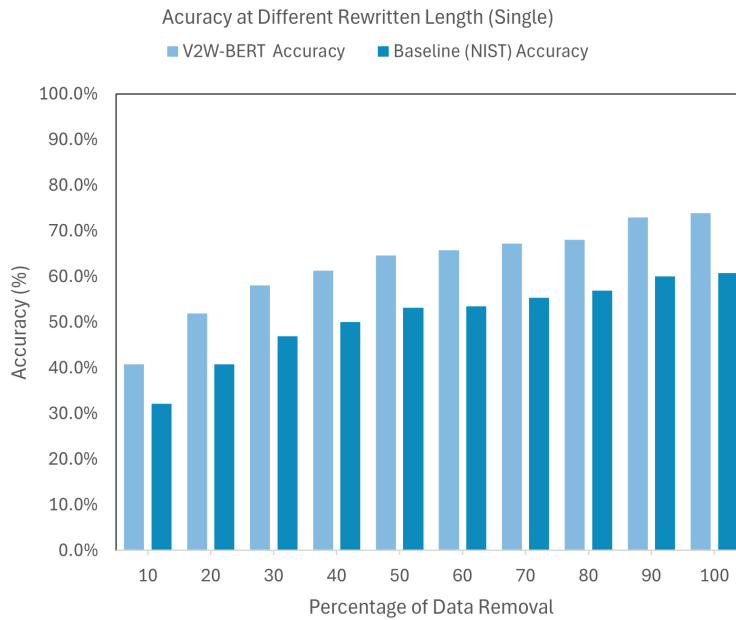


Figure 1.13: Accuracy at Different Rewritten Length Percentages (single prediction)

Results. The results are summarized in Table ?? and Figures 1.13 and 1.14, showing the accuracy of GPT-4o predictions under varying lengths of rewriting, prediction types, and baselines. Two types of predictions were evaluated: Single Prediction and Top-3 Predictions, and their performance was compared against two baselines: NIST and V2W-BERT. The text lengths range from 10% to 100% of the original description.

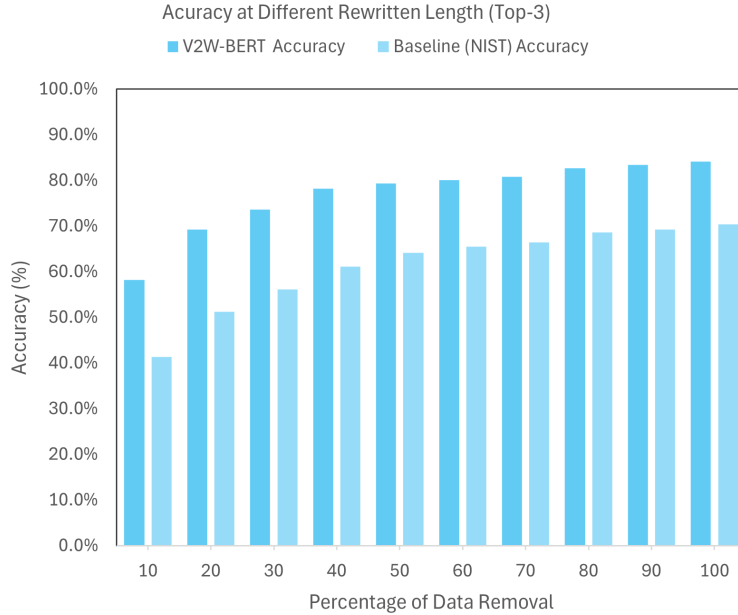


Figure 1.14: Accuracy at Different Rewritten Length Percentages (top 3 prediction)

The results highlight the following trends: (i) Accuracy improves as the length of the rewritten text increases, regardless of the prediction type or baseline. (ii) The Top-3 predictions consistently outperform Single Prediction across all lengths, indicating that providing multiple predictions enhances performance. (iii) The V2W-BERT baseline achieves higher accuracy than the Baseline (NIST) across all text lengths, demonstrating its robustness for mapping CWE IDs.

Observations. We observe that the accuracy of GPT-4o varied significantly with the length of the input text. Shorter descriptions (e.g., 10%, 40%) often led to reduced accuracy, as key contextual information was lost. However, rewritten texts in the range of 50%-70% of the original length achieved a balance between brevity and content, resulting in accuracy levels similar to the

Prediction Type	Length	Baseline (NIST)	V2W-BERT
Single Prediction	10%	32.2%	40.8%
	20%	40.8%	51.9%
	30%	47.0%	58.1%
	40%	50.1%	61.3%
	50%	53.1%	64.6%
	60%	55.3%	65.7%
	70%	55.3%	67.2%
	80%	57.0%	68.1%
	90%	60.8%	72.0%
	100%	60.8%	73.9%
Top-3 Predictions	10%	41.4%	58.2%
	20%	51.2%	69.9%
	30%	56.1%	73.6%
	40%	61.1%	78.2%
	50%	64.1%	79.0%
	60%	65.4%	80.8%
	70%	66.4%	81.8%
	80%	67.9%	82.8%
	90%	69.3%	83.4%
	100%	70.4%	84.2%

original descriptions.

As expected, GPT-4o performed best when provided with full descriptions, retaining all context and detail for optimal accuracy. For comparisons with V2W-BERT, our results suggest that the predictions from V2W-BERT align more closely with GPT-4o predictions, achieving higher accuracy rates compared to the Baseline (NIST) CWE data as presented in Table ???. The maximum accuracy achieved with GPT-4o occurred when using three predictions, with V2W-BERT mapping as the baseline comparison.

In some specific cases, using 10% of the description, GPT was able to generate a set of CWE IDs, with the clarification that more information is needed to obtain a better prediction. These are some of the specific cases when

the text size was around 10%:

Example 1: *“I’m sorry, but I need more specific details about the vulnerability in the Totolink LR1200GB to determine the most likely CWE weaknesses. If you provide additional information on the nature of the vulnerability, I may be able to help identify the possible CWE weaknesses.”*

Example 2: *“Without specific details on the type of vulnerability present in the Totolink LR1200GB, it’s difficult to pinpoint the exact CWEs. However, typically common vulnerabilities associated with routers like these could be: CWE-20, CWE-787, CWE-79”*

Unseen CVEs. The performance of GPT4-o was evaluated on recently published CVEs. A total of 250 samples, were used for testing. The number of correct Single Predictions was 182, with an accuracy of 72.80%, and the number of correct Top-3 Predictions was 209, with an accuracy of 83.60%.

1.7. Conclusion

We introduced generative AI tools to enable cyber defenders to efficiently traverse through siloed datasets for vulnerabilities, weaknesses, attack patterns, attack mechanisms, and mitigation actions to gain insights and make appropriate decisions. By building on large language models, we presented techniques to map information, which can then be modeled as a knowledge graph. By combining recently developed retrieval augmented generation (RAG) methodologies with cyber knowledge graphs, we demonstrated excellent accuracies for conversational style access to information retrieval. We believe that the work presented here lays the foundation for both human and automated cyberde-

fense.

While this work presented state-of-the-art methodologies in applying generative AI tools for cyberdefense, there is considerable scope for advancement. Specific directions for future research come from further enhancing the accuracy of mapping vulnerabilities, weaknesses, attack patterns/mechanisms, and mitigation actions in a holistic manner through advanced language models; augmenting optimization tools for prioritizing mitigative actions for a given cybersystem; automating defense approaches by combining the proposed approaches with deep reinforcement learning frameworks; and, developing comprehensive validation and verification techniques. We believe that the rapidly growing threats to cybersystems can be effectively countered by sophisticated AI-enabled cyberdefense techniques.

Bibliography

Ehsan Aghaei, Xi Niu, Waseem Shadid, and Ehab Al-Shaer. SecureBERT: A domain-specific language model for cybersecurity. In *International Conference on Security and Privacy in Communication Systems*, pages 39–56. Springer, 2022.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024a.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762, 2024b.

Siddhartha Shankar Das, Edoardo Serra, Mahantesh Halappanavar, Alex Pothen, and Ehab Al-Shaer. V2W-BERT: A framework for effective hierarchical multiclass classification of software vulnerabilities. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–12. IEEE, 2021.

Siddhartha Shankar Das, Ashutosh Dutta, Sumit Purohit, Edoardo Serra, Mahantesh Halappanavar, and Alex Pothen. Towards automatic mapping of vulnerabilities to attack patterns using large language models. In *2022 IEEE*

International Symposium on Technologies for Homeland Security (HST), pages 1–7, 2022a. doi: 10.1109/HST56032.2022.10025459.

Siddhartha Shankar Das, Mahantesh Halappanavar, Antonino Tumeo, Edoardo Serra, Alex Pothén, and Ehab Al-Shaer. Vwc-bert: Scaling vulnerability–weakness–exploit mapping on modern ai accelerators. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 1224–1229, 2022b. doi: 10.1109/BigData55660.2022.10020622.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '24*, page 6491–6501, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671470.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*, 2024.

Mohamed Manzour Hussien, Angie Nataly Melo, Augusto Luis Ballardini, Carlota Salinas Maldonado, Rubén Izquierdo, and Miguel Angel Sotelo. Rag-based explainable prediction of road users behaviors for automated driving using knowledge graphs and large language models. *Expert Systems with Applications*, 265:125914, 2025.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications.

- IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2022. doi: 10.1109/TNNLS.2021.3070843.
- Yan Jia, Yulu Qi, Huaijun Shang, Rong Jiang, and Aiping Li. A practical approach to constructing a knowledge graph for cybersecurity. *Engineering*, 4(1):53–60, 2018.
- Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.
- Mukesh Kumar, Utsav Anand Mani, Pranjal Tripathi, Mohd Saalim, and Sneha Roy. Artificial hallucinations by google bard: Think before you leap. *Cureus*, 15, 2023.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- Kai Liu, Fei Wang, Zhaoyun Ding, Sheng Liang, Zhengfei Yu, and Yun Zhou. Recent progress of using knowledge graph for cybersecurity. *Electronics*, 11(15):2287, 2022a.
- Kai Liu, Fei Wang, Zhaoyun Ding, Sheng Liang, Zhengfei Yu, and Yun Zhou. A review of knowledge graph application scenarios in cyber security. *arXiv preprint arXiv:2204.04769*, 2022b.

Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. Creating cybersecurity knowledge graphs from malware after action reports. *IEEE Access*, 8:211691–211703, 2020.

Tyler Thomas Procko and Omar Ochoa. Graph retrieval-augmented generation for large language models: A survey. In *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)*, pages 166–169. IEEE, 2024.

Moqsadur Rahman, Krish O. Piryani, Aaron M. Sanchez, Sai Munikoti, Luis De La Torre, Maxwell S. Levin, Monika Akbar, Mahmud Hossain, Monowar Hasan, and Mahantesh Halappanavar. Retrieval augmented generation for robust cyber defense. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (United States), 09 2024. URL <https://www.osti.gov/biblio/2474934>.

Diego Sanmartin. Kg-rag: Bridging the gap between knowledge and creativity. *arXiv preprint arXiv:2405.12035*, 2024.

Guowei Shen, Wanling Wang, Qilin Mu, Yanhong Pu, Ya Qin, and Miao Yu. Data-driven cybersecurity knowledge graph construction for industrial control system security. *Wireless Communications and Mobile Computing*, 2020 (1):8883696, 2020.

Leslie F Sikos. Cybersecurity knowledge graphs. *Knowledge and Information Systems*, 65(9):3511–3531, 2023.

Karthik Soman, Peter W Rose, John H Morris, Rabia E Akbas, Brett Smith, Braian Peetoom, Catalina Villouta-Reyes, Gabriel Ceron, Yongmei Shi, Angela Rizk-Jackson, et al. Biomedical knowledge graph-enhanced prompt generation for large language models. *arXiv preprint arXiv:2311.17330*, 2023.

Yujie Sun, Dongfang Sheng, Zihan Zhou, and Yifei Wu. AI hallucination: towards a comprehensive classification of distorted information in artificial intelligence-generated content. *Palgrave Communications*, 11(1):1–14, December 2024. doi: 10.1057/s41599-024-03811-.

Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox, and Jonas Partner. *Neofj in Action*. Manning, 2015. ISBN 1617290769 9781617290763.

Zhentao Xu, Mark Jerome Cruz, Matthew Guevara, Tie Wang, Manasi Deshpande, Xiaofeng Wang, and Zheng Li. Retrieval-augmented generation with knowledge graphs for customer service question answering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2905–2909, 2024.

Chengshuai Zhao, Garima Agrawal, Tharindu Kumarage, Zhen Tan, Yuli Deng, Ying-Chih Chen, and Huan Liu. Ontology-aware rag for improved question-answering in cybersecurity education. *arXiv preprint arXiv:2412.14191*, 2024a.

Xiaojuan Zhao, Rong Jiang, Yue Han, Aiping Li, and Zhichao Peng. A survey on cybersecurity knowledge graph construction. *Computers & Security*, 136: 103524, 2024b.