CVE-TTP KG: KNOWLEDGE GRAPH LINKING SOFTWARE VULNERABILITIES TO ATTACK BEHAVIORS

Swati Yadav
Department of Computer Science & Engineering
Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur
swati.yadav@skit.ac.in

Dincy R. Arikkat
Department of Computer Science and Engineering
Christ College of Engineering Thrissur, Kerala
dincyrarikkat.cse@cce.edu.in

Basant Agarwal
Department of Computer Science & Engineering
Central University of Rajasthan, Ajmer
basant@curaj.ac.in

Serena Nicolazzo
Department of Electrical, Computer and Biomedical Engineering,
University of Pavia, Italy
serena.nicolazzo@unipv.it

Antonino Nocera
Department of Electrical, Computer and Biomedical Engineering,
University of Pavia, Italy
antonino.nocera@unipv.it

Vinod P.
Department of Computer Applications,
Cochin University of Science
and Technology, India
vinod.p@cusat.ac.in

ABSTRACT

In the evolving threat landscape, adversaries exploit software vulnerabilities to launch sophisticated attacks, challenging traditional defenses. Although databases like CVE and NVD provide detailed technical information, they often lack links to attacker behaviors such as tactics and techniques, limiting effective threat interpretation and response. This work bridges this gap by connecting vulnerabilities with behavioral patterns from the MITRE ATT&CK framework. We construct a CVETTP Knowledge Graph that links CVEs to tactics and techniques using classification and relation extraction. Transformer-based models are developed for behavior identification, with CySecBERT achieving macro F1-scores of 87.71% (techniques) and 96.16% (tactics). Also, we created an annotated dataset with $24,820$ entities and $43,608$ relations for entity and relation extraction. The pipeline-based approach achieves macro F1-scores of 0.86 (entity extraction) and 0.99 (relation extraction), while a span-based joint model achieves 0.78. These outputs are integrated into a Neo4j-based Cyber Threat Knowledge Graph, enabling structured visualization of vulnerabilities.

***K*eywords** Cyber Threat Intelligence, Knowledge Graph, MITRE ATT&CK, Vulnerability Analysis, Joint Entity Relation Extraction.

1 Introduction

In the contemporary cybersecurity landscape, software vulnerabilities have become strategic assets that are actively traded, weaponized, and integrated into coordinated, economically motivated cyber campaigns by threat actors [4]. Each disclosed vulnerability represents a potential entry point into critical systems and is often exploited as part of broader attack operations [13]. To standardize the reporting of such flaws, repositories like the Common Vulnerabilities

and Exposures (CVE) [1] and the National Vulnerability Database (NVD)[2] have been developed. Although these repositories offer structured identifiers and metadata for known vulnerabilities, they frequently lack operational context, such as attacker objectives, techniques, and exploitation paths needed to understand how vulnerabilities are actually used in real-world threat scenarios. Without this context, security teams are left with large volumes of vulnerability data but limited insight into which flaws are most likely to be exploited and how.

Traditional defensive strategies focus on patching and mitigation [6]. However, with the increasing sophistication of attacks, technical fixes alone are insufficient. To achieve cyber resilience, defenders must understand the intent behind adversary behavior and how specific vulnerabilities are linked to strategic goals. Frameworks like MITRE ATT&CK [3] provide structured knowledge of Tactics, Techniques, and Procedures (TTPs), where *tactics* represent the attackers goals or objectives (the *what*), and *techniques* describe the general methods used to achieve those goals (the *how*). Integrating behavioral intelligence with vulnerability data is essential for prioritizing threats and anticipating attack patterns based on adversary intent and capability. The urgency of this need is magnified by the rapid growth in vulnerability disclosures. In 2024 alone, over 40,000 new CVEs were reported, marking a 39% increase from the previous year[4]. This explosion of data makes manual analysis difficult and necessitates automated solutions that can process, correlate, and contextualize vulnerabilities at scale[5].

To support contextualized vulnerability analysis, we propose a CVE-TTP Knowledge Graph (CVE-TTP KG) that links software vulnerabilities to attacker behaviors, specifically, the tactics and techniques. Our system first identifies relevant ATT&CK tactics and techniques associated with the vulnerabilities using security-focused transformed models. Furthermore, the framework supports the extraction of key vulnerability entities and their relationships in the form of semantic triples: *<entity1, relation, entity2 >*. These triples serve as the foundational input for constructing the CVE-TTP KG.

This KG addresses a critical gap in current vulnerability analysis frameworks, which often treat vulnerabilities as isolated records lacking behavioral context. By explicitly modeling the connections between CVEs and adversary behaviors, our approach enables a deeper understanding of exploitation patterns, threat objectives, and attack chains. The CVE-TTP KG not only supports visualization and situational awareness for human analysts but also facilitates machine reasoning for automated threat detection, risk prioritization, and decision-making. The primary contributions of this work are outlined as follows:

- We introduce novel datasets for multi-label classification of ATT&CK techniques and tactics, and a manually annotated dataset of vulnerability entities with semantic relations to support knowledge extraction and graph-based security analysis.
- We implement a transformer-based classification model to automatically map software vulnerabilities to relevant ATT&CK tactics and techniques.
- We develop models for extracting entities and their semantic relationships from vulnerability descriptions using both pipeline and joint learning approaches. We conduct a comparative evaluation of these two strategies to assess their effectiveness in capturing vulnerability-specific relations.
- We construct a comprehensive KG that links CVEs to associated attack behaviors along with other relevant vulnerability information. The resulting graph offers a structured and interpretable representation of threat intelligence, enabling more effective analysis of vulnerability exploitation across adversarial campaigns.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 details the proposed architecture and methodology. Section 4 presents the experimental results along with a comprehensive analysis. Finally, Section 5 concludes the study and outlines directions for future research.

2 Related Work

This section reviews relevant previous work under two main themes: (1) CVE-to-TTP classification, and (2) Threat KG construction for vulnerability-centric threat modeling.

[1] https://cve.mitre.org/
[2] https://nvd.nist.gov/
[3] https://attack.mitre.org/
[4] https://socradar.io/top-10-exploited-vulnerabilities-of-2024/
[5] https://www.bitsight.com/blog/2025-predictions-for-cve-vulnerabilities

2.1 CVE-TTP Classification

Recent years have seen growing efforts to map CVEs to MITRE tactics and techniques. Simonetto et al. [17] proposed a rule-based pipeline that leverages structured data such as CWE and semantic similarity to align CVEs with techniques. However, their approach relied on classical machine learning and lacked automation and contextual reasoning using modern NLP techniques. Kuppa et al. [12] introduced a deep learning-based model that uses pseudo-labeled data to train on semantic matches between CVE text and ATT&CK technique descriptions. Although effective in technique prediction, their work did not incorporate tactic-level classification or structured threat modeling.

Recent advances in using Large Language Models (LLMs) have opened new avenues. Hu et al. [11] presented LLM-TIKG, a prompting-based framework that enhanced annotation and classification in cybersecurity tasks. However, their approach required manual intervention and was not tailored to CVE descriptions. In contrast, our work enables end-to-end, automated multi-label classification of both tactics and techniques and releases annotated datasets for benchmarking. Zhang et al. [22] proposed UniTTP, a unified framework for the extraction of TTP across various cybersecurity inputs using prompt-based encoder-decoder models. While their system handles joint reasoning across sources such as Indicators of Compromises (IoCs) and reports, it lacks a specific focus on CVE-driven threat modeling.

2.2 Threat Knowledge Graph Construction

Constructing effective threat KGs depends on accurately identifying relevant entities and their relationships within cybersecurity data. Several works have constructed cybersecurity KGs from open-source or structured threat intelligence. Xiao et al. [21] explored embedding-based models to detect relationships among software entities, yet did not target ATT&CK TTP semantics or vulnerability-to-technique links. Shi et al. [16] and Falcarin et al. [8] integrated heterogeneous data sources into graph-based representations but lacked direct extraction from CVEs or a focus on MITRE TTPs.

Gao et al. [9] introduced ThreatKG, an automated KG generation system from CTI reports using semantic enrichment. While rich in contextual information, their model does not specifically connect CVEs to ATT&CK frameworks. Similarly, Hu et al. [11] built AttackKG using LLM-driven joint NER and RE on CTI reports, with limited attention to CVE alignment. Other notable efforts include CSKG4APT by Ren et al. [14], focused on attribution of APT campaigns via integration of malware and TTPs, and OSTIS by Arikkat et al. [1], which constructed organizational threat KGs. Though both frameworks support advanced threat modeling, they do not explicitly extract or link CVE descriptions to ATT&CK techniques. Zhang et al. [23] proposed AttacKG+, combining LLMs with automated CTI parsing to generate ATT&CK-aligned graphs. Their methodology shares similarities with ours in leveraging LLMs, though they primarily operate on threat reports. Our contribution bridges this gap by applying joint entity-relation extraction directly on CVEs and constructing ATT&CK-grounded graphs for structured visualization of adversarial behavior.

Existing approaches primarily focus either on CVE-to-TTP mapping or KG construction, with limited efforts toward integrating both in a unified framework. They also lack joint NER and relation extraction tailored to CVE texts and provide limited support for ATT&CK-based visualization of adversarial behavior. To address these gaps, we propose an integrated framework that performs multi-label classification of ATT&CK tactics and techniques from CVE descriptions, applies joint entity and relation extraction for enriched threat intelligence, and constructs a vulnerability-centric KG for structured representation and visualization of adversarial behavior.

3 Methodology

This section outlines the architecture of the proposed CVE-TTP KG system, as illustrated in Figure 1. The ultimate goal of our system is to construct a comprehensive KG by transforming extracted CVE descriptions into a curated, semantically enriched format designed to capture adversarial behavior patterns. The workflow is structured into four primary modules: *(i)* Data Collection, *(ii)* Linking CVE descriptions with techniques and tactics, *(iii)* Entity and Relation Extraction, *(iv)* KG Generation. Each module is responsible for a specific task within the pipeline.

3.1 Data Collection

Initially, we constructed a CVE dataset covering the period 1999–2025 using records from the NVD and the MITRE CVE List V5. NVD provided structured CVE records from 2002 onward, while CVE List V5[6] was used to supplement the years 1999–2001 and ensure historical completeness. After data collection, a total of 276,289 CVE records were

[6] `https://github.com/CVEProject/cvelistV5`

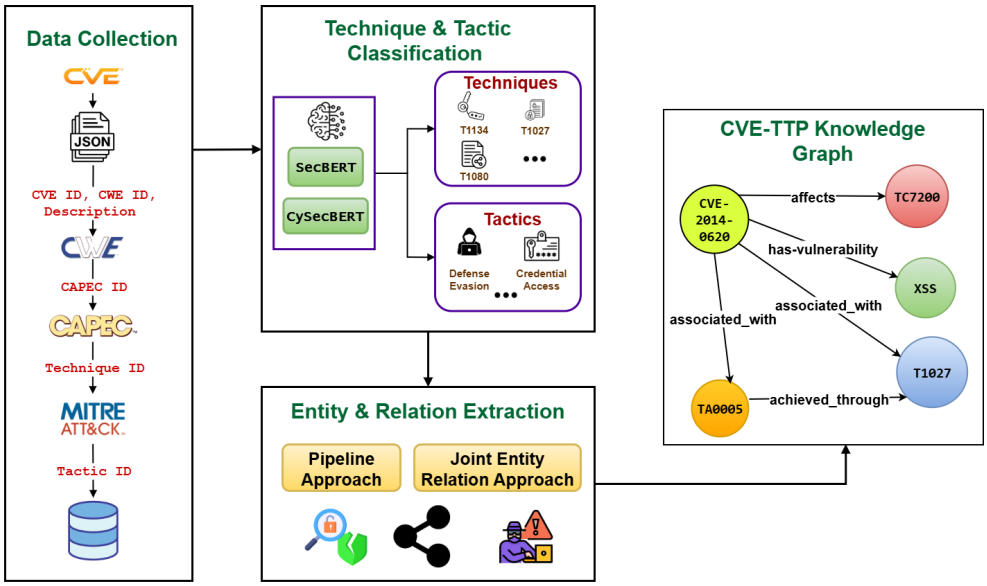

Figure 1: Architecture of the CVE-TTP Knowledge Graph. The construction process consists of four stages: *(i)* data collection, where vulnerability information is gathered from multiple sources; *(ii)* classification, where CVEs are mapped to ATT&CK techniques and tactics using a prediction model; *(iii)* entity and relation extraction, which identifies structured information; and *(iv)* knowledge graph construction, where the extracted triples are organized into the graph.

obtained. Figure 2 presents the year-wise distribution of collected CVEs, showing a steady increase in reported vulnerabilities from 1999 to 2024.

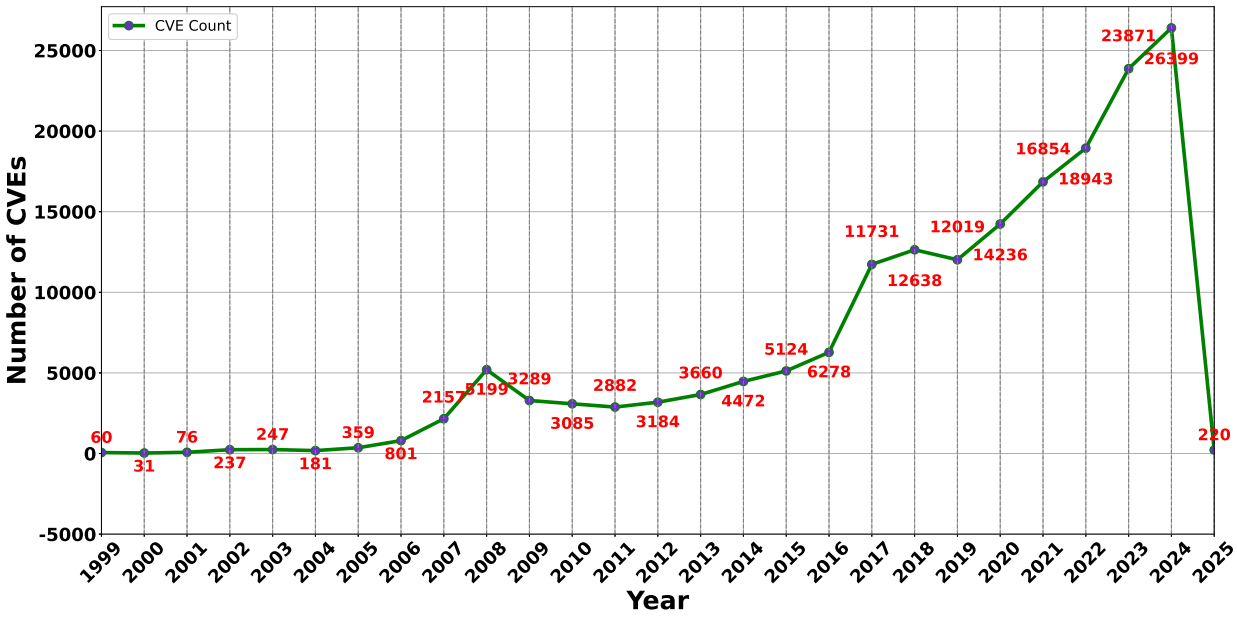

Figure 2: Year-wise distribution of CVEs. The plot depicts the annual distribution of collected vulnerabilities, showing the number of CVE entries extracted for each year.

To enrich CVEs with adversarial behavior, we integrated information from CWE, CAPEC, and the MITRE ATT&CK framework using a multi-stage mapping process. First, CWE identifiers were extracted from NVD records. For exam-

ple, CVE-2021-22909[7], a Man-in-the-Middle vulnerability in EdgeMAX EdgeRouter firmware updates, is associated with CWE-300 (Channel Accessible by Non-Endpoint)[8]. This CWE maps to CAPEC attack patterns such as CAPEC-94 (Adversary in the Middle) and CAPEC-57[9], which further align with ATT&CK Technique T1040 (Network Sniffing)[10]. The technique is linked to the Credential Access and Discovery tactics.

This enrichment pipeline (CVE → CWE → CAPEC → Technique → Tactic) was used to generate ground-truth labels for classification. CVEs without valid CWE mappings were excluded. Of the 276,289 collected CVEs, 178,233 were successfully enriched, resulting in a technique dataset covering 95 ATT&CK techniques and a tactic dataset spanning all 14 ATT&CK tactic categories. This approach forms a solid foundation but suffers from inherent limitations due to the sparse and incomplete mappings available across public resources, particularly from CWE to CAPEC to ATT&CK techniques. As a result, this method enriches only a subset of CVEs with behavioral context. To overcome this limitation, we developed an automated classification model that predicts ATT&CK tactics and techniques directly from CVE textual descriptions.

3.2 Linking CVE description with Techniques and Tactics

In this phase, we developed an automated tool based on transformer-based models to predict the techniques and tactics associated with a given vulnerability description. This model leverages the CVE dataset constructed in the previous phase (Session 3.1). Specifically, each input instance to the model was formulated by combining key contextual information, including the CVE identifier, its textual description, and the associated CWE identifier. The input format was structured as: *"The vulnerability identified as CVE ID: <CVE ID >is described as: <Description >. It is associated with CWE ID: <Problem Type (CWE) >."*

This dataset was then used to train multi-label classification models for predicting adversarial behaviours. To perform technique and tactic classification, we employed pretrained BERT-based language models that have been specifically adapted to the cybersecurity domain. **SecBERT** is a BERT model pretrained on cybersecurity corpora such as APTnotes, STUCCO-Data, CASIE, and SecureNLP, and we used the jackaduma/SecBERT version from Hugging Face. **CySecBERT**[2] is a cybersecurity-adapted BERT model pretrained on blogs, arXiv, NVD, and Twitter data, and we used markusbayer/CySecBERT for fine-tuning on our dataset.

3.3 Entity and Relation Extraction

In this phase, we identify and extract the cybersecurity-specific entities and their semantic relationships, particularly those related to software vulnerabilities. We defined the entities and relationships by reviewing prior works [12, 18, 16] on software vulnerability entity and relation extraction. Furthermore, to ensure consistency with established standards in CTI, the entity-relation schema is aligned with the structural specifications of the STIX 2.0 framework [11]. The identified entities include vulnerability identifiers such as CVE IDs and CWE IDs, Product Name, Product Version, Vendor Name, Vulnerability Type, and Impact. To align with adversarial behavior modeling, the schema also includes Tactic and Technique entities.

Based on these entities, several key relationships are defined to capture meaningful connections within the KG. The *affects* relationship links a vulnerability to a specific product or hardware it impacts (e.g., CVE-2021-29529 affects TensorFlow). The *has_version* relationship associates a product with its specific version (e.g., TensorFlow has version TensorFlow 2.4.1), while *has_vendor* identifies the organization responsible for the product (e.g., TensorFlow has vendor Google). The *has_weakness* relationship connects a vulnerability or product to a corresponding CWE category (e.g., CVE-2021-29529 has weakness CWE-131). The *has_impact* relationship represents the effect of a vulnerability on security properties such as confidentiality, integrity, or availability. Furthermore, the *associated_with* relationship captures links between vulnerabilities and adversarial techniques or between weaknesses and their related CWE categories (e.g., CVE-2021-29529 associated with T1021, and Buffer Overflow associated with CWE-120). The *related_to* relationship is used to denote connections between similar or linked vulnerabilities (e.g., CVE-2009-2879 related to CVE-2009-2876). Finally, the *achieved_through* relationship links a tactic to the technique used to accomplish it (e.g., TA0002 achieved through T1204).

[7] https://www.cve.org/CVERecord?id=CVE-2021-22909

[8] https://cwe.mitre.org/data/definitions/300.html

[9] https://capec.mitre.org/data/definitions/57.html

[10] https://attack.mitre.org/techniques/T1040/

[11] https://oasis-open.github.io/cti-documentation/stix/intro.html

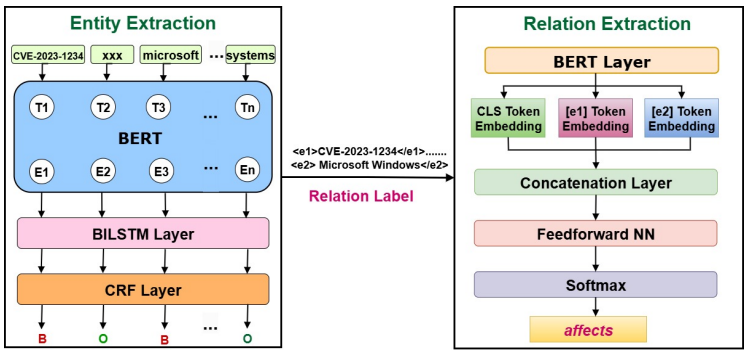

Figure 3: Workflow of pipeline-based entity and relation extraction

After defining the target entity categories and relation types, we manually annotated and constructed a gold-standard dataset for entity and relation extraction using *Label Studio*[12]. Due to the time-intensive nature of manual annotation, we selected a representative sample of $1,080$ descriptions from the $178,233$ CVE records, based on a 95% confidence level and a 3% margin of error. To perform entity and relation extraction, we experimented with both pipeline-based and joint extraction approaches. The pipeline method first identifies entities, which are then passed to a separate relation extraction model to determine the semantic links between them. In contrast, the joint extraction approach simultaneously detects entities and their relations, constructing relational triples in the form <*subject, relation, object* >.

For the pipeline approach, entity recognition is formulated as a BIO tagging task, where each token is labeled as B (beginning), I (inside), or O (outside) of an entity. For example, in "CVE-2023-1234 affects Microsoft Windows systems", "CVE-2023-1234" is tagged as B-CVE_ID, while "Microsoft Windows" is tagged as B-PRODUCT and I-PRODUCT. For the relation extraction stage, the two target entities in each instance (e_1, e_2) were highlighted in the input sentence by surrounding e_1 with the special marker "$" and e_2 with "#".

For the joint extraction model, the input is represented in the SciERC format, which consists of: *(i)* `tokens` — tokenized CVE descriptions; *(ii)* `entities` — labeled spans with type and index positions; and *(iii)* `relations` — semantic links between entity pairs. The resulting labeled dataset consists of $1,080$ CVE descriptions, encompassing $24,820$ entities and $43,608$ relations for training and evaluating entity and relation extraction models.

3.3.1 Pipeline-based Entity and Relation Extraction:

In this approach, entity and relation extraction are performed sequentially through a two-stage: *(i)* Cybersecurity Entity Recognition (CER) and *(ii)* Relation Extraction (RE). The CER stage is designed to identify cybersecurity-related entities from vulnerability (CVE) descriptions, while the RE stage focuses on uncovering semantic associations between these entities. The workflow of the pipeline-based entity and relation extraction is depicted in Figure 3.

Cybersecurity Entity Recognition: In the CER stage, we adopt a BERT–BiLSTM–CRF architecture, which has been widely recognized for its effectiveness in domain-specific sequence labeling tasks [1, 3, 19, 20]. BERT (`bert-base-uncased`) is used as the encoder to generate contextual token embeddings that capture semantic and technical information. These embeddings pass through a BiLSTM layer to model sequence context in both directions, followed by a CRF layer that ensures consistent label sequences and improves recognition of ambiguous or multi-token cybersecurity entities.

Relation Extraction: In the RE stage, we identify the relations between entities. We adopted a transformer-based `bert-base-uncased` classifier, motivated by the demonstrated dominance of BERT-based methods in achieving state-of-the-art performance for relation extraction across diverse domains [5]. To guide the models attention toward the target entities e_1 and e_2, we enclose them with special markers ("$" for e_1 and "#" for e_2) in the input sentence. The annotated text is then encoded by BERT into contextualized hidden representations. Entity span embeddings are derived via pooling, concatenated, and fed into a fully connected layer with a softmax activation function to predict the relation type.

[12] `https://labelstud.io/`

3.3.2 Joint Entity and Relation Extraction:

Traditional pipeline architectures for entity and relation extraction suffer from error propagation and fail to capture the mutual dependencies between entities and their relations [10]. To address these limitations, we investigated a joint extraction framework that performs both tasks jointly. Specifically, we utilize a span-based transformer framework [7] that processes CVE vulnerability descriptions to extract entity-relation triples in the format $(\texttt{subject}, \texttt{relation}, \texttt{object})$. The overall workflow of the joint entity and relation extraction process is illustrated in Figure 4.

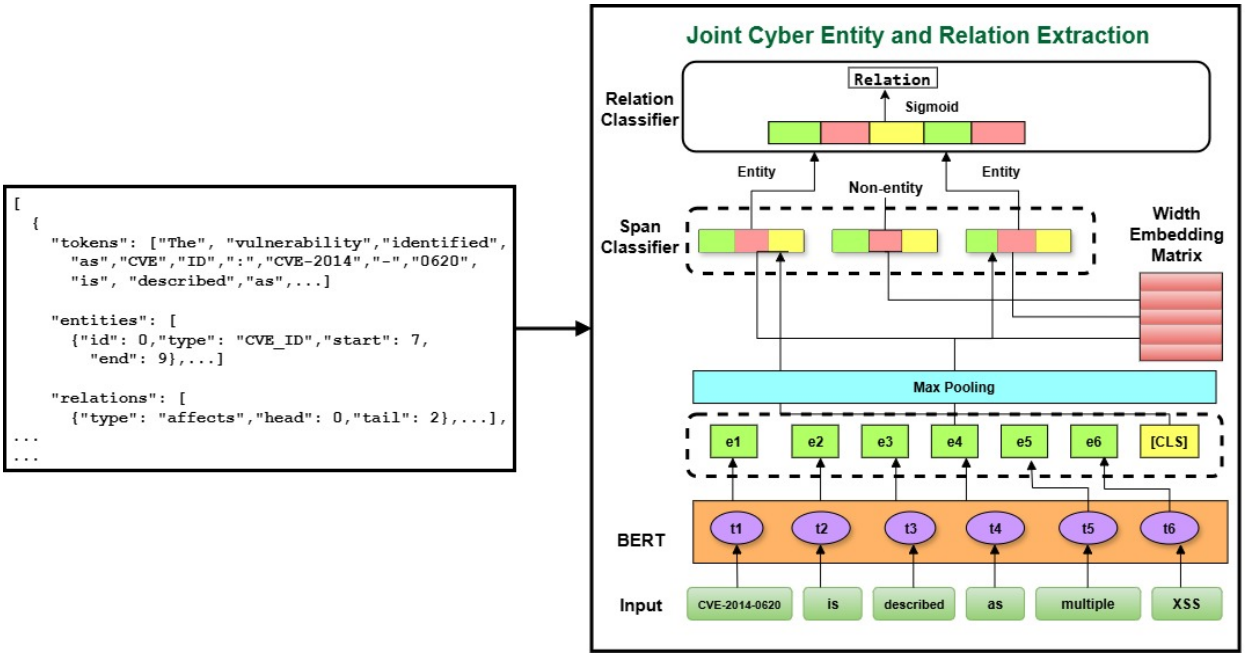

Figure 4: Workflow for joint entity and relation extraction

The core architecture of the joint entity and relation extraction model is a pre-trained CySecBERT. Instead of token-level labeling, the model operates on spans (contiguous sequences of tokens), treating each possible span as a candidate for entity classification. This method avoids traditional decoding strategies like CRFs or BIO tagging. The joint entity model operates in three stages: *(i)* span classification, *(ii)* span filtering, and *(iii)* relation classification.

Entity Identification via Span Classification: For each sentence, the model generates contextual embeddings for all tokens, including a special `[CLS]`(classifier) token that represents the overall context of the sentence. Candidate spans are constructed from consecutive token sequences, each of which is encoded using max pooling over its token embeddings. To enrich the span representation, two additional vectors are concatenated: *(i)* a width embedding that captures the length of the span, and *(ii)* the `[CLS]` embedding to integrate sentence-level information. This concatenated vector is passed through a softmax classifier to predict an entity type or a non-entity class.

Span Filtering: Spans predicted as non-entities are filtered out. To reduce computational overhead, we discard spans exceeding a predefined maximum length before classification. The resulting set comprises only those spans that are likely to represent meaningful entities.

Relation Classification: Each pair of retained entity spans is evaluated to determine whether a semantic relation exists between them. For each ordered pair (s_1, s_2), a composite vector is formed by concatenating: *(i)* the span embeddings of s_1 and s_2, *(ii)* their respective width embeddings, and *(iii)* a context embedding derived from the token embeddings found between the two spans.

If the span pair is adjacent or overlapped, the context vector is replaced by a zero vector. This combined representation is fed into a sigmoid classifier to determine the presence of one or more relations from a predefined set. Since relations can be directional, both (s_1, s_2) and (s_2, s_1) are evaluated.

3.4 CVE-TTP Knowledge Graph Construction

A Cybersecurity KG (CKG) provides a structured and semantically rich representation of cybersecurity-related concepts. In this framework, key entities such as vulnerabilities, threat actors, exploits, attack tactics, techniques, etc. are modeled as nodes, while the semantic relationships among them are represented as directed edges. This KG provides a unified structure that bridges low-level technical vulnerabilities with high-level adversarial strategies. The construction of the CVE-TTP KG involves the following key stages:

- **Entity and Relation Extraction:** Entities and their semantic relationships are extracted from CVE vulnerability descriptions using the extraction methods described in the Section 3.3.
- **Graph Generation using Neo4j:** The extracted entity-relation triples are used to populate a CVE-TTP graph using Neo4j [13]. Neo4j, a native graph database, enables efficient storage, traversal, and querying of the graph structure using the Cypher query language. This facilitates downstream analytics such as subgraph exploration, threat attribution, vulnerability prioritization, and adversary behavior mapping. Each unique entity is represented as a node with associated metadata (e.g., type, name, identifier), and each relation is stored as a directed edge connecting two nodes, labeled with the corresponding relation type.

4 Experiment & Analysis

In this section, we detail the experimental setup, followed by a discussion of the results and their analysis.

4.1 Experimental Setup

The experimentation was conducted using Python on a local machine powered by an Apple M3 chipset. The system featured an 8-core integrated GPU and leveraged Apples Metal 3 API for hardware acceleration. We employed libraries, including PyTorch and Hugging Face Transformers for model training and inference, scikit-learn for performance evaluation, matplotlib for result visualization, and the Neo4j Python driver for constructing the KG.

To evaluate the performance of the proposed model across different tasks, we employed metrics appropriate to each problem. Since the classification of tactics and techniques involves predicting multiple labels per instance, we adopted evaluation metrics commonly used in multi-label classification. These include Hamming Loss and Jaccard Similarity, along with standard metrics including Precision, Recall, F1-Score, and Accuracy [15]. For entity recognition and relation extraction tasks, we evaluated model performance using Precision, Recall, F1-Score, and Accuracy to assess the quality and correctness of the extracted entities and their semantic relations.

4.2 Performance on Technique and Tactic Classification

This section presents the results of the proposed model for technique and tactic classification from vulnerability descriptions. For the experimentation, we have used the labeled dataset generated through the process outlined in Section 3.1. The technique classification dataset contains 178,233 vulnerability descriptions annotated with 95 unique techniques, while the tactic classification dataset includes the same 178,233 descriptions labeled with 14 unique tactics. To ensure the reliability of supervised learning, we excluded labels with insufficient samples. Specifically, techniques and tactics with fewer than 50 instances were discarded. After this filtering, the final dataset used for model training includes 80 techniques and 13 tactics. The dataset was split into 80:10:10 for training, validation, and testing. For technique and tactic prediction, we used pre-trained cybersecurity models, SecBERT and CySecBERT, fine-tuned on our dataset with five random seeds for stability. Experiments used the AdamW optimizer with a learning rate of 1×10^{-5} for up to five epochs. Batch sizes were set to 8 for `CySecBERT` and 16 for `SecBERT` based on GPU constraints. Binary Cross-Entropy with Logits was used to handle the multi-label classification task.

The performance on the test set for both technique and tactic classification is summarized in Table 1. Overall, CySecBERT consistently outperforms SecBERT in both tasks.In the technique classification task, CySecBERT achieves a higher Jaccard score (0.9659) and F1-macro (0.8771). Similarly, Hamming loss is lower for CySecBERT (0.0071). For the tactic classification task, both models achieve even higher performance across all metrics, with CySecBERT again outperforming SecBERT. Specifically, CySecBERT reaches a Jaccard score of 0.9858, F1-micro of 0.9899, and F1-macro of 0.9616, reflecting generalization across all tactic labels.

[13] https://neo4j.com/

Table 1: Average performance of SecBERT and CySecBERT models for technique and tactic classification, averaged over five different random seed runs.

	Technique Classification		Tactic Classification	
Metric (Avg.)	**SecBERT**	**CySecBERT**	**SecBERT**	**CySecBERT**
Jaccard Score	0.9584	0.9659	0.9829	0.9858
Hamming Loss	0.0085	0.0071	0.0084	0.0071
F1 Micro	0.9582	0.9657	0.9879	0.9899
F1 Macro	0.8361	0.8771	0.9530	0.9616
F1 Weighted	0.9574	0.9651	0.9877	0.9898
Precision Micro	0.9749	0.9797	0.9920	0.9929
Precision Macro	0.9199	0.9334	0.9823	0.9849
Precision Weighted	0.9744	0.9794	0.9919	0.9928
Recall Micro	0.9421	0.9517	0.9839	0.9869
Recall Macro	0.7927	0.8381	0.9281	0.9413
Recall Weighted	0.9421	0.9517	0.9839	0.9869
Accuracy	0.9159	0.9379	0.9461	0.9552

4.3 Performance on Entity and Relation Extraction

We utilized the labeled dataset constructed through the steps outlined in Section 3.3. The dataset comprises $1,080$ CVE vulnerability descriptions with $24,820$ entities and $43,608$ relations. The dataset was divided into training, validation, and test sets with an 80:10:10 proportion.

4.3.1 Pipeline Approach:

In the first stage of our pipeline approach, we extracted the entities using BERT–BiLSTM–CRF architecture. The `bert-base-cased` encoder first generates contextual embeddings, which are passed through a BiLSTM with a hidden size of 256, a fully connected layer, and a CRF layer for tag prediction. Training was carried out for 10 epochs with the AdamW optimizer, a maximum sequence length of 256 tokens, a learning rate of 2×10^{-5}, a batch size of 16, and a dropout rate of 0.1. The proposed CER module achieved an F1 micro score of 0.98, an F1 weighted score of 0.99, and an F1 macro score of 0.86 on the test set. The second stage identifies semantic relations between extracted entity pairs using `bert-base-uncased` with a maximum sequence length of 512 tokens. The model was trained for one epoch using Adam, a learning rate of 1×10^{-5}, batch size 16, and dropout 0.2 before each dense layer. The relation extraction (RE) system in the pipeline approach achieved an F1-score of 0.99, with a precision of 0.9961 and a recall of 0.9960. The confusion matrices for the CER and RE stages are presented in Appendix Figure 7 and Figure 8.

4.3.2 Joint Entity and Relation Extraction Approach:

For joint entity and relation extraction, we used a span-based EntityRelation Transformer [7] with CySecBERT. The model was trained for 25 epochs with a batch size of 2, using a learning rate of $2e-5$, linear warmup of 0.1, and weight decay of 0.01. Gradient clipping (1.0) and dropout (0.2) were applied, with a maximum span size of 10. To balance samples, we used 300 negative entities and 200 negative relations per batch, limiting candidate span pairs to 750 per document and applying a relation threshold of 0.5. A span size embedding of 25 was included, and training was parallelized using four processes. Early stopping was based on validation performance. For entity extraction, the joint model achieved a micro-F1 score of 95.86 and a macro-F1 score of 88.20.

Furthermore, we evaluated the relation extraction performance of the joint entityrelation model under two settings: the NEC (Named Entity Classification) constraint and the non-NEC (span-only) condition. In the NEC setting, evaluation is stringent, requiring that each predicted relation instance satisfy three criteria: (i) both entity spans are correctly identified, (ii) the entity types are accurately classified, and (iii) the predicted relation label matches the gold standard. In contrast, the non-NEC setting (also referred to as a "relaxed" evaluation) requires only that the entity spans and the relation label are correct, without enforcing the correctness of entity type assignments. Under these settings, the joint model achieved a micro-F1 of 77.81 and a macro-F1 of 78.70 in the non-NEC scenario, and a micro-F1 of 77.77 with a macro-F1 of 78.69 in the NEC scenario. The detailed class-wise performance of entities and relations is presented as a confusion matrix in Appendix Figure 9 and Figure 10.

4.4 CVE-TTP Knowledge Graph Generation

The construction of the CVE-TTP KG transforms outputs from entity and relation extraction models into structured triples, which are then imported into Neo4j for visualization and analysis. For the joint extraction model, predictions are stored in JSON format, containing tokens, identified entities, relations, and their spans. These outputs are programmatically converted into $\langle$*head, relation, tail*$\rangle$ triples to semantically represent connections between entities. In contrast, the pipeline model produces triples composed of the *head*, *relation*, and *tail* without span information, since entity and relation predictions are generated in separate sequential steps. For the KG generation task, we employed the

same CVE descriptions to assess the performance of both the pipeline and joint models. Specifically, we used 50 vulnerability descriptions that were not included in the training phase of any model to generate triples. For demonstration purposes, we illustrate the process using the CVE-2020-0617 vulnerability:

> The vulnerability identified as CVE ID: `CVE-2020-0617` is described as: A denial of service vulnerability exists when Microsoft Hyper-V Virtual PCI on a host server fails to properly validate input from a privileged user on a guest operating system, also known as "Hyper-V Denial of Service Vulnerability". It is associated with `CWE ID: CWE-20`. For organizations monitoring cyber threat activities, this vulnerability aligns with Tactic ID(s): `TA0003, TA0004, TA0005, TA0006` and Technique ID(s): `T1027, T1036.001, T1539, T1553.002, T1562.003, T1574.006, T1574.007`, representing a critical threat to security measures and warranting immediate attention.

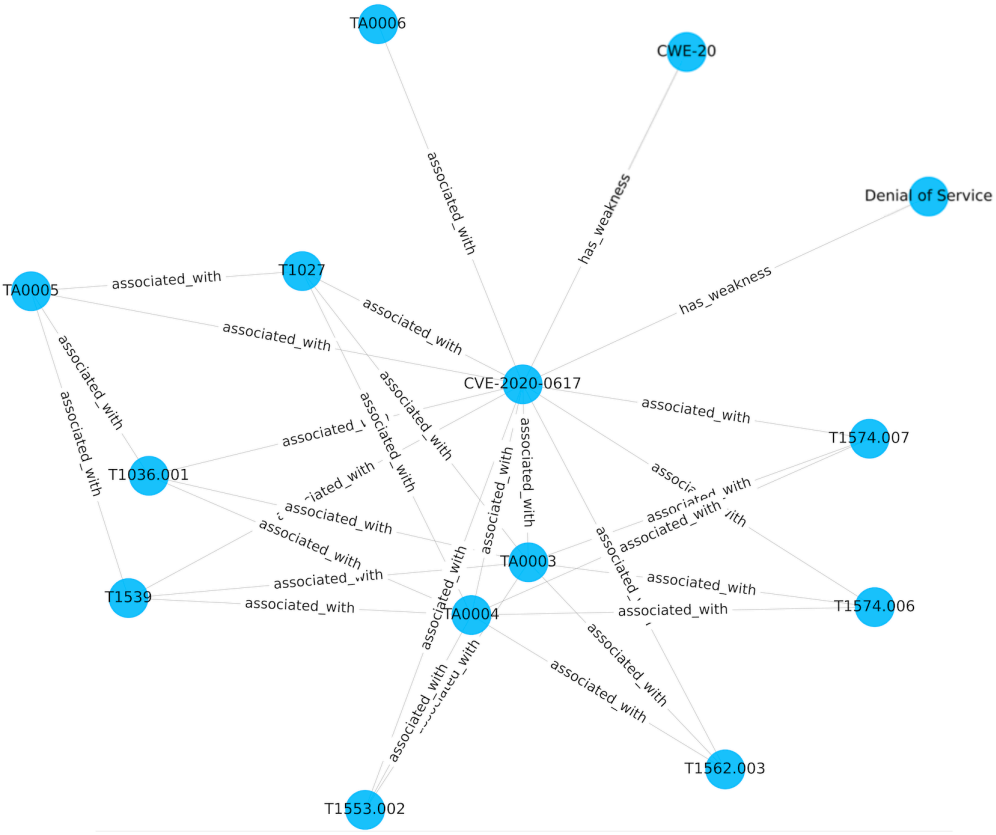

Figure 5: CVE-TTP KG generated by the pipeline model, illustrating entity associations alongside semantic inconsistencies such as generic tactic-technique relations.

Pipeline Model KG Generation: The ⟨*head, relation, tail*⟩ triples extracted using CER and relation extraction are imported into Neo4j for KG construction and visualization. For the CVE-2020-0617 vulnerability, the pipeline model's KG includes (Figure 5) key entities such as `CVE-2020-0617` (vulnerability identifier), `CWE-20` (weakness classification), multiple tactic nodes (`TA0005`, `TA0003`, `TA0004`, `TA0006`), and technique nodes (`T1027`, `T1036.001`, `T1539`, `T1553.002`, `T1562.003`, `T1574.006`, `T1574.007`). The relation `associated_with` predominates, linking tactics and techniques to the central CVE node. However, this model often replaces the semantically accurate `achieved_through` relation, which specifies how tactics are operationalized through techniques, with the more generic `associated_with` relation. This substitution reduces the granularity and semantic precision in mapping procedural cybersecurity workflows. Furthermore, while the pipeline model correctly predicts the `has_weakness` relation between CVE and CWE entities, it misclassifies "Denial of Service" as a weakness rather than properly categorizing it as a vulnerability type. This misclassification adversely affects the accuracy of representing vulnerability impacts, unlike the joint model, which successfully captures this context.

Joint Model KG Generation: Alternatively, the joint extraction model performs entity recognition and relation extraction concurrently, predicting ⟨*head, relation, tail*⟩ triples directly from text without intermediate entity pairing. This end-to-end inference reduces error propagation found in pipeline architectures and enhances extraction efficiency. The resulting triples were imported into Neo4j for KG construction. The resulting KG centers on `CVE-2020-0617`

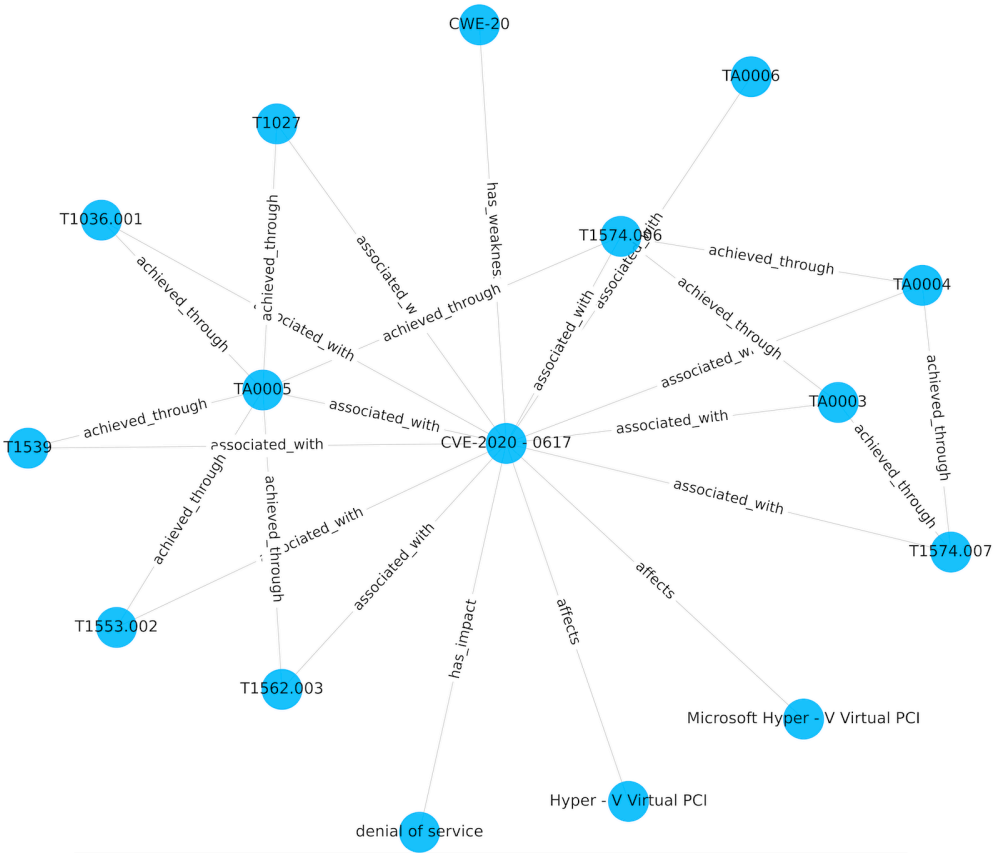

Figure 6: CVE-TTP KG generated by the joint model includes CVE, CWE, product, tactic, and technique entities with semantically precise relations.

(Figure 6), representing a denial of service vulnerability in Microsoft Hyper-V Virtual PCI stemming from input validation flaws. This node connects to entities such as `CWE-20` (weakness category), related products (`Microsoft Hyper-V Virtual PCI`), and the impact type (`denial of service`). The graph captures extensive security context through tactic nodes (`TA0005`, `TA0003`) and technique nodes (`T1027`, `T1539`, `T1533.002`), reflecting mappings aligned to MITRE adversarial patterns.

Relations extracted include `associated_with`, linking CVEs to tactics and techniques; `achieved_through`, specifying how tactics are executed via techniques or sub-techniques; `has_weakness`, mapping CVEs to weakness classifications; `affects`, connecting vulnerabilities to impacted products; and `has_impact`, representing the consequence of the vulnerability. These relations form a semantically rich and technically precise structure emphasizing attack sequences and vulnerability impacts, albeit with less representation of vendor or product version contexts. Compared to the pipeline KG, the joint model produces a more compact and focused KG, with strong CVE-to-CWE and CVE-to-TTP edges, but fewer explicit tactic-to-technique connections. This outcome highlights a trade-off: the joint model excels in end-to-end robust extraction of core vulnerability and weakness mappings while sacrificing some granularity in operational semantics concerning attack workflows. While the joint model generally predicts `None` when uncertain, it also exhibits misclassifications between semantically similar entity types. For example, both models failed to correctly identify the `Vendor` entity "Microsoft" and the associated relation `has_vendor` ⟨`Hyper-V Virtual PCI`, `has_vendor`, `Microsoft`⟩. The pipeline model omitted this entity and relation entirely, whereas the joint model incorrectly classified "Microsoft Hyper-V Virtual PCI" as a `Product`. This behavior reflects the joint model's integrated prediction mechanism, which strives to assign the most plausible labels jointly under ambiguous contexts, contrasting with the pipeline model's tendency to omit difficult predictions. Despite these misclassifications, the joint model's unified approach mitigates error propagation between entity and relation extraction subtasks, a common limitation in the pipeline design. Consequently, the joint model demonstrates superior performance and robustness for comprehensive KG generation, especially in contexts with interdependent and overlapping entity-relation semantics.

5 Conclusion

The rapid increase in software vulnerabilities challenges cybersecurity teams, especially in linking vulnerabilities to real-world attack behaviors. Although databases like CVE and NVD provide technical details, they lack connections to tactics and techniques, limiting effective threat response. In this work, we propose CVE-TTP KG, a framework that extracts cybersecurity entities and relationships from CVE descriptions to construct a structured KG to improve threat analysis. We collected and mapped data linking CVEs with weaknesses, tactics, and techniques, and created two datasets for multi-label classification and entityrelation extraction. Using CySecBERT, we achieved macro F1-scores of 0.8771 for technique classification and 0.9616 for tactic classification. For extraction, the pipeline approach reached 0.86 (entities) and 0.99 (relations), while the span-based model achieved 0.78. The resulting KG provides a clear representation of vulnerabilityattack relationships, supporting better analysis and mitigation. Future work includes integrating Large Language Models to improve extraction and addressing challenges like entity disambiguation and coreference resolution. Resolving ambiguous terms (e.g., "service") and references (e.g., "it" or "this issue") will enhance entity linking and improve the accuracy and usefulness of the KG for threat modeling.

References

[1] Arikkat, D.R., Vinod, P., KA, R.R., Nicolazzo, S., Nocera, A., Timpau, G., Conti, M.: Ostis: A novel organization-specific threat intelligence system. Computers & Security **145**, 103990 (2024)

[2] Bayer, M., Kuehn, P., Shanehsaz, R., Reuter, C.: Cysecbert: A domain-adapted language model for the cybersecurity domain. ACM Transactions on Privacy and Security **27**(2), 1–20 (2024)

[3] Dasgupta, S., Piplai, A., Kotal, A., Joshi, A.: A comparative study of deep learning based named entity recognition algorithms for cybersecurity. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 2596–2604. IEEE (2020)

[4] Di Tizio, G., Armellini, M., Massacci, F.: Software updates strategies: A quantitative evaluation against advanced persistent threats. IEEE Transactions on Software Engineering **49**(3), 1359–1373 (2022)

[5] Diaz-Garcia, J.A., Lopez, J.A.D.: A survey on cutting-edge relation extraction techniques based on language models. Artificial Intelligence Review **58**(9), 287 (2025)

[6] Dissanayake, N., Jayatilaka, A., Zahedi, M., Babar, M.A.: Software security patch management-a systematic literature review of challenges, approaches, tools and practices. Information and Software Technology **144**, 106771 (2022)

[7] Eberts, M., Ulges, A.: Span-based joint entity and relation extraction with transformer pre-training. arXiv preprint arXiv:1909.07755 (2019)

[8] Falcarin, P., Dainese, F.: Building a cybersecurity knowledge graph with cybergraph. In: Proceedings of the 2024 ACM/IEEE 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS) and 2024 IEEE/ACM Second International Workshop on Software Vulnerability. pp. 29–36 (2024)

[9] Gao, P., Liu, X., Choi, E., Ma, S., Yang, X., Song, D.: Threatkg: An ai-powered system for automated open-source cyber threat intelligence gathering and management (2024), `https://arxiv.org/abs/2212.10388`

[10] Guo, Y., Liu, Z., Huang, C., Liu, J., Jing, W., Wang, Z., Wang, Y.: Cyberrel: Joint entity and relation extraction for cybersecurity concepts. In: Information and Communications Security: 23rd International Conference, ICICS 2021, Chongqing, China, November 19-21, 2021, Proceedings, Part I 23. pp. 447–463. Springer (2021)

[11] Hu, Y., Zou, F., Han, J., Sun, X., Wang, Y.: Llm-tikg: Threat intelligence knowledge graph construction utilizing large language model. Computers & Security **145**, 103999 (2024)

[12] Kuppa, A., Aouad, L., Le-Khac, N.A.: Linking cves to mitre att&ck techniques. In: Proceedings of the 16th International Conference on Availability, Reliability and Security. pp. 1–12 (2021)

[13] Makrakis, G.M., Kolias, C., Kambourakis, G., Rieger, C., Benjamin, J.: Vulnerabilities and attacks against industrial control systems and critical infrastructures. arXiv preprint arXiv:2109.03945 (2021)

[14] Ren, Y., Xiao, Y., Zhou, Y., Zhang, Z., Tian, Z.: Cskg4apt: A cybersecurity knowledge graph for advanced persistent threat organization attribution. IEEE Transactions on Knowledge and Data Engineering **35**(6), 5695–5709 (2022)

[15] Riera, T.S., Higuera, J.R.B., Higuera, J.B., Herraiz, J.J.M., Montalvo, J.A.S.: A new multi-label dataset for web attacks capec classification using machine learning techniques. Computers & Security **120**, 102788 (2022)

[16] Shi, Z., Matyunin, N., Graffi, K., Starobinski, D.: Uncovering cwe-cve-cpe relations with threat knowledge graphs. ACM Transactions on Privacy and Security **27**(1), 1–26 (2024)

[17] Simonetto, S., Bosch, P.: Comprehensive threat analysis and systematic mapping of cves to mitre framework. In: 1st International Conference on Natural Language Processing and Artificial Intelligence for Cyber Security, NLPAICS 2024 (2024)

[18] Sun, J., Xing, Z., Lu, Q., Xu, X., Zhu, L.: A multi-faceted vulnerability searching website powered by aspect-level vulnerability knowledge graph. In: 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). pp. 60–63. IEEE (2023)

[19] Wang, X., He, S., Xiong, Z., Wei, X., Jiang, Z., Chen, S., Jiang, J.: Aptner: A specific dataset for ner missions in cyber threat intelligence field. In: 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD). pp. 1233–1238. IEEE (2022)

[20] Wang, X., Liu, R., Yang, J., Chen, R., Ling, Z., Yang, P., Zhang, K.: Cyber threat intelligence entity extraction based on deep learning and field knowledge engineering. In: 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD). pp. 406–413. IEEE (2022)

[21] Xiao, H., Xing, Z., Li, X., Guo, H.: Embedding and predicting software security entity relationships: A knowledge graph based approach. In: Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part III 26. pp. 50–63. Springer (2019)

[22] Zhang, J., Wen, H., Li, L., Zhu, H.: Unittp: A unified framework for tactics, techniques, and procedures mapping in cyber threats. In: 2024 IEEE 23rd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 1580–1588. IEEE (2024)

[23] Zhang, Y., Du, T., Ma, Y., Wang, X., Xie, Y., Yang, G., Lu, Y., Chang, E.C.: Attackg+:boosting attack knowledge graph construction with large language models (2024), `https://arxiv.org/abs/2405.04753`

A Class-wise Analysis of Entity Relation Extraction Approaches

A.1 Pipeline Approach

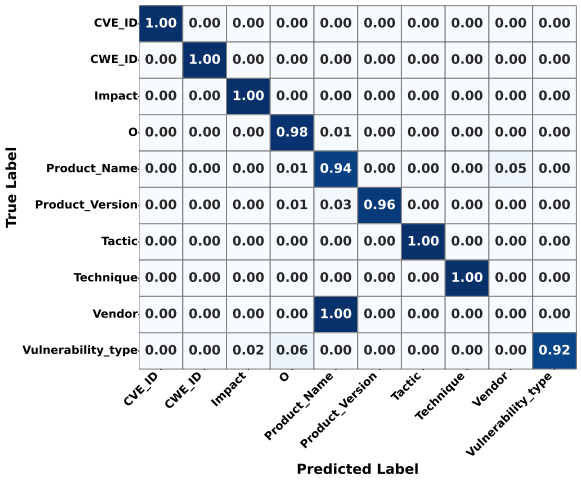

Figure 7: Confusion matrix for the Entity Recognition stage of the pipeline approach.

The confusion matrix for the CER phase is presented in Figure 7. From the confusion matrix, we can infer that several entities, including `CVE_ID`, `CWE_ID`, `Impact`, `Tactic`, and `Technique`, achieved a classification performance of 1.00. Conversely, the model's performance decreases for categories exhibiting greater semantic overlap. In particular, `Vendor` entity is misclassified as `Product_Name`. This confusion likely stems from the lexical and contextual similarity between vendor names and product identifiers (e.g., *Microsoft* or *Oracle* can occur as both vendor and product in CVE text). Overall, the CER module demonstrates strong performance across most entity types. Most misclassifications involve low-frequency classes with overlapping contextual characteristics.

The individual relation type performance of the RE task is presented in Figure 8. Several relation types, including `achieved_through`, `affects`, `has_vendor`, and `related_to`, are predicted without error (score of 1.00). The most pronounced challenge emerged with `has_impact`, achieving 0.95 accuracy, with around 5% of cases re-labelled

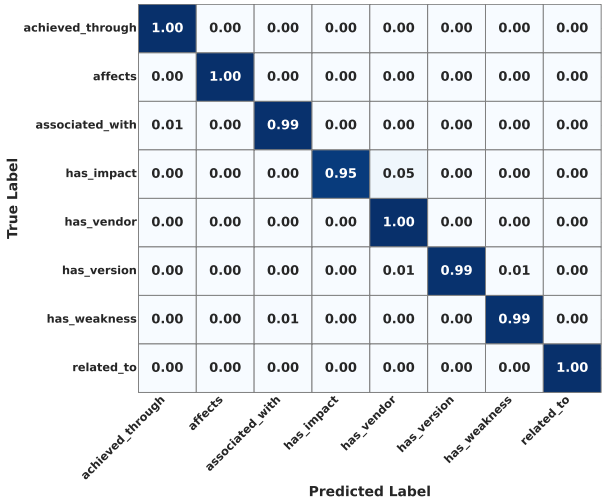

Figure 8: Confusion matrix for the Relation Extraction stage of the pipeline approach.

as `has_vendor`. This confusion likely stems from the narrative style of vulnerability descriptions, although the RE component achieved high performance across relation types.

A.2 Joint Entity and Relation Extraction Approach

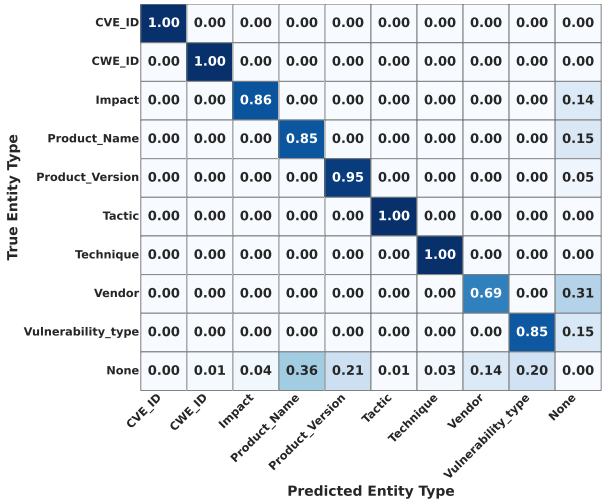

Figure 9: Confusion matrix for entities in the joint model.

Figure 10 presents the confusion matrix of the entities in the joint model. The results indicate that the model performs exceptionally well on structured identifiers and well-defined categories, including `CVE_ID`, `CWE_ID`, `Tactic`, and `Technique`, with perfect performance of 1.00, and 0.95 for `Product_Version`. In contrast, the `Vendor` class exhibits comparatively lower performance (0.69), with approximately 31% of instances predicted as `None`. In the confusion matrices for the joint entity and relation extraction model, the label `None` signifies that the model predicted the absence of a valid entity or relation.

The confusion matrix shown in Figure 10 summarizes the models performance on the relations in the joint model. The results indicate strong performance for relation types such as `has_weakness` (1.00), `related_to` (1.00), `associated_with` (0.99), and `achieved_through` (0.96). For the `has_vendor` relation, a reduced performance of 0.53 was observed. The lower score for `has_vendor` is mainly due to weaker `Vendor` entity predictions, which limit correct relation extraction. In particular, 47% of `has_vendor` instances were misclassified as `None`, showing difficulty in learning this low-frequency relation.

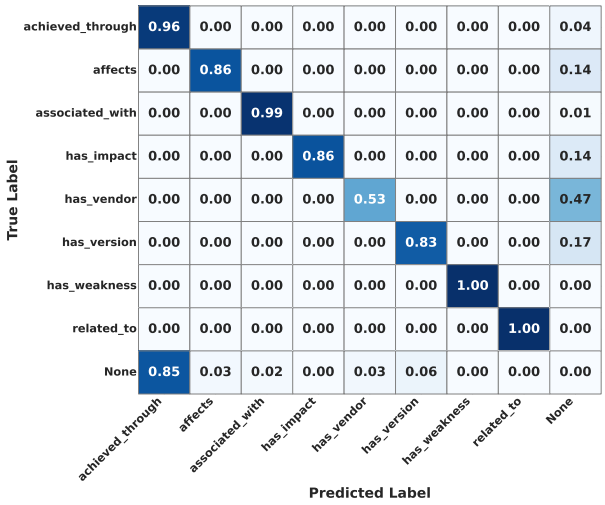

Figure 10: Confusion matrix for relations in the joint model.