A Data-Driven Ontology Framework for Integrating Heterogeneous Vulnerability Sources

Domenico Amalfitano[1,†], Domenico Benfenati[1,*,†], Davide Landolfi[1,*,†], Antonio Maria Rinaldi[1], Cristiano Russo[1] and Cristian Tommasino[1]

[1]*Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Via Claudio, 21, Naples, 80125, Italy*

Abstract

The increasing complexity of cyber threats, combined with the dispersion of information across multiple independent data sources, underscores the need for a structured, formal, and interoperable representation of vulnerability-related knowledge. Existing datasets, although comprehensive within their scopes, often lack semantic alignment and explicit connections between key entities such as vulnerabilities, weaknesses, and attack patterns. In this work, we present an enhanced ontology-based framework for vulnerability knowledge representation that systematically improves upon existing theoretical models through data-driven design principles. Our ontology model address critical inconsistencies between theoretical frameworks and real-world vulnerability databases by conducting a comprehensive analysis of actual data structures from CVE, CWE, CAPEC, and CPE repositories. We demonstrate the practical applicability of our framework through the implementation of a multi-model NoSQL database populated with real vulnerability data and validate its effectiveness using SWRL-based inference rules that derive new knowledge from existing relationships. The resulting knowledge base enables automated reasoning about attack chains, vulnerability exploitation patterns, and security relationships, providing a foundation for advanced cybersecurity analysis.

Keywords

Cybersecurity, Vulnerability, Ontological Database, NIST, MITRE

1. Introduction

The continuous evolution of cyber threats has underscored the need for structured, interoperable vulnerability datasets [1]. Key security repositories include CVE [2], CWE [3], CPE [4], and CAPEC [5]. Our work uses data from the National Vulnerability Database (NVD) [6], which enriches the foundational CVE list maintained by MITRE [7] with additional metadata and CVSS [8] severity scores.

Despite their richness, these repositories exhibit significant heterogeneity in data formats and classification schemas, preventing comprehensive integrated analysis [9, 10]. CVE and CWE use JSON APIs or XML [11] with varying structures, CAPEC employs XML with a distinct taxonomy, and CVSS follows a unique scoring syntax. This fragmentation limits security analysts' ability to understand complete attack landscapes and assess cascading risks [12, 13].

To address this gap, we present a practical ontology model that bridges heterogeneous vulnerability sources through systematic adaptation of existing ontological frameworks [14] to real-world database constraints. Our contribution transforms established cybersecurity ontologies [15, 16] into database-compliant schemas that preserve semantic relationships while accommodating actual field structures and referential constraints in CVE, CWE, CPE, and CAPEC repositories.

Proceedings of the Joint Ontology Workshops (JOWO) - Episode XI: The Sicilian Summer under the Etna, co-located with the 15th International Conference on Formal Ontology in Information Systems (FOIS 2025), September 8-9, 2025, Catania, Italy

*Corresponding authors.

†These authors contributed equally.

domenico.amalfitano@unina.it (D. Amalfitano); domenico.benfenati@unina.it (D. Benfenati); dav.landolfi@studenti.unina.it (D. Landolfi); antoniomaria.rinaldi@unina.it (A. M. Rinaldi); cristiano.russo@unina.it (C. Russo); cristian.tommasino@unina.it (C. Tommasino)

0000-0002-4761-4443 (D. Amalfitano); 0009-0008-5825-8043 (D. Benfenati); 0000-0001-7003-4781 (A. M. Rinaldi); 0000-0002-8732-1733 (C. Russo); 0000-0001-9763-8745 (C. Tommasino)

2. Related Work

The use of ontologies in vulnerability management and security risk assessment has matured from early efforts focused on general-purpose security models [17] to more refined approaches tailored to the representation and reasoning of vulnerability-specific data [15, 14]. A central goal in this domain has been the establishment of standardized, interoperable schemas that consolidate information from structured vulnerability repositories such as CVE, CWE, and CAPEC [14]. The ontologies in this field often aim to formalize relationships among vulnerabilities, weaknesses, and attack patterns, allowing automated reasoning over causal chains and shared characteristics. In particular, inference rules have been introduced to support the identification of consequences, causal dependencies, and congeneric relationships between vulnerabilities [14], facilitating a deeper understanding of their systemic impact. Alongside this structural modeling, some approaches have extended the scope of ontology to include contextual entities such as IT products, threats, countermeasures, and even external intelligence sources like social media [16, 18], and also for obfuscation in the crawling phase [19]. This broader perspective improves the ontology's ability to represent real-world attack surfaces and supports more comprehensive risk assessments. Standards from organizations such as NIST and CVSS are frequently adopted to ensure alignment with established security frameworks [16, 17], promoting compatibility with existing tools and methodologies.

Despite these advancements, a number of limitations remain. In many cases, product-specific semantics and real-time threat context are not sufficiently modeled [15], limiting the applicability of these ontologies in dynamic environments. Furthermore, while inference mechanisms are increasingly integrated, their expressiveness and scalability in large, heterogeneous systems are still underexplored. Overall, the body of work reflects a steady progression toward more expressive and interoperable knowledge models, but there remains a need for deeper integration of dynamic, product-level, and cross-domain information to fully support operational cybersecurity decision making.

3. Ontology Model Design

In this work we improve the general top ontology model proposed by [14] to better represent and integrate real repositories related to software vulnerability. In particular, our ontological model enhances the framework through systematic evaluation of its coherence with real-world vulnerability databases. We identified critical inconsistencies between the theoretical model and actual data structures: certain ontological properties were absent from real data sources, while important database attributes lacked theoretical representation. Our enhanced model addresses these limitations, ensuring full coherence with actual database schemas.

We defined four core entity types—`Vulnerability`, `Weakness`, `Attack Pattern`, and `Product`—corresponding to CVE, CWE, CAPEC, and CPE standards. Properties were systematically selected based on conceptual importance and actual database presence. Relationships are defined through object properties that capture semantic connections in real datasets, such as `hasWeakness` linking vulnerabilities to associated weaknesses with exact relationship metadata found in CVE-CWE mappings. Our complete model structure is shown in Figure 1, and accessible in this Zenodo repository[1].

We included a separate `CWE Category` class for category-level weaknesses, which represent broader groupings distinct from individual CWE entries. While NIST discouraged CVE-to-category mappings since 2019 and these account for $< 0.1\%$ of records, we retain them to distinguish entries with less granular information from fully described weaknesses with complete technical attributes.

3.1. CVE Class

Our enhanced `Vulnerability` class improves upon the ontological model of [14] through systematic analysis of real data. Key enhancements address critical limitations in the original framework:

[1] https://doi.org/10.5281/zenodo.15915548

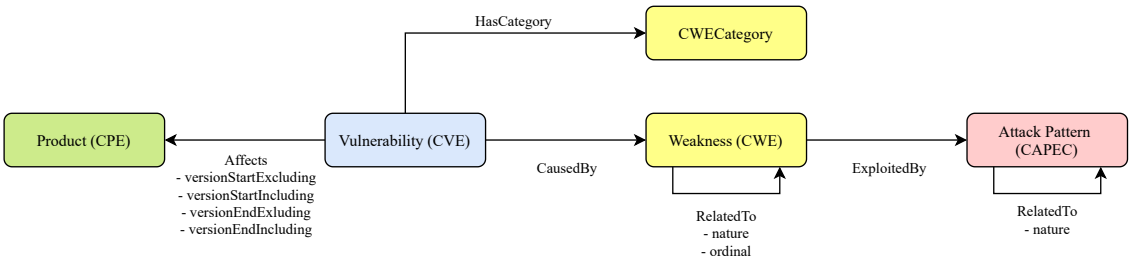

Figure 1: Diagram of the enhanced ontological model classes and their relationships.

CVSS Metrics Integration: While [14] fragmented CVSS data into separate attributes, we consolidated all CVSS parameters into a single, cohesive property maintaining semantic integrity and better alignment with NIST data schemas [8, 6].

Enhanced References Structure: Our model extends the original URL-only references by adding a "tags" sub-property that classifies reference types, improving semantic reasoning capabilities for vulnerability analysis [2].

Product Relationship Refinement: Unlike the original model, we integrate direct relationships with CPE through a separate product class, enabling comprehensive tracking of affected products [4]. Our CVE-CPE relationship includes metadata about specific version ranges, constituting a semantically rich association rather than simple linking.

Figure 2 shown the class diagram of the Vulnerability class. In grey we indicate the multiple sub-attributes of the class, and in white the simple attributes.

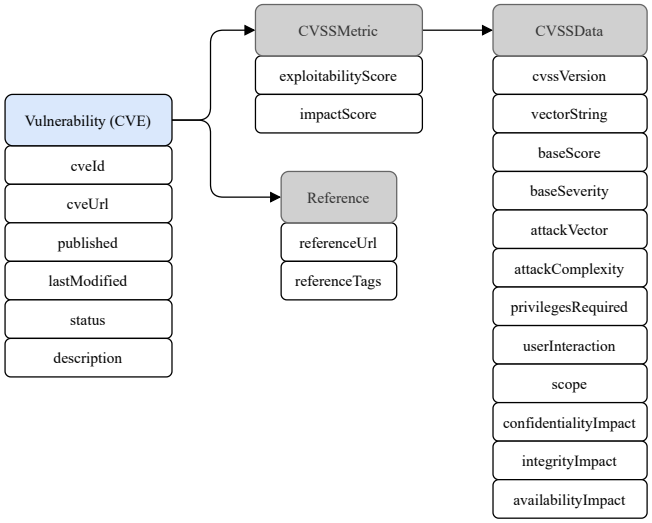

Figure 2: Class diagram illustrating the core CVE entity and its associated components in the proposed vulnerability ontology model.

3.2. CWE Class

The CWE class represents software and system weaknesses at varying levels of abstraction, capturing both their nature and the contexts in which they can be encountered or exploited. We enhance and refine the vulnerability ontology model through data-driven design principles, focusing exclusively on fields that are present in the official MITRE CWE data sources. This enhancement ensures that our improved ontology accurately reflects the available information while maintaining practical applicability for real-world vulnerability analysis.

We refine the framework in [14] and deliberately exclude CWSS (Common Weakness Scoring System) components from the CWE class definition. This design decision stems from the fact that MITRE does not provide CWSS scores in their official data feeds, and while these scores can be calculated post-hoc, they are not inherently present in the source data. The list of attributes present in the CWE data is shown in Figure 3. We decided to maintain the `likelihoodOfExploit` property as it appears directly in the source data as an attribute of the CWE class. However, since the official data contains no information linking to CWSS calculations or scoring mechanisms, we exclude any CWSS-related properties from our CWE class definition. This approach ensures that our ontology remains faithful to the actual data structure while avoiding assumptions about unavailable information.

Among the comprehensive set of available CWE properties, we focus here on the most critical attributes for vulnerability analysis and ontology-driven inference:

- **description:** A concise textual summary of the weakness, describing its fundamental characteristics and the types of software flaws or design omissions it encompasses. This description provides essential context for understanding how the weakness arises and what classes of vulnerabilities it may enable.
- **commonConsequences:** A structured list of potential negative outcomes that may follow exploitation of the weakness. Each entry comprises:
  - **Scope:** the specific security property (e.g. Confidentiality, Integrity, Availability) that the exploitation violates;
  - **Impact (optional):** a technical description of the effect should an adversary successfully leverage the weakness;
  - **Likelihood (optional):** an ordinal indication of how probable each consequence is relative to others, facilitating risk-based prioritization when multiple impact scenarios exist.
- **potentialMitigations:** A set of recommended countermeasures or design strategies designed to prevent or reduce the severity of the weakness. Each mitigation entry includes:
  - **Phase:** the software development lifecycle stage (e.g. design, implementation, testing) at which the mitigation is most effectively applied;
  - **Strategy:** a high-level approach or best practice that guides the realization of the mitigation (e.g. input validation, coding standards, runtime checks);
  - **Effectiveness:** a brief assessment of how well the mitigation addresses the weakness, including any known limitations;
  - **Description:** detailed commentary on the mitigation's application, benefits, and potential trade-offs.
- **observedExamples:** References to concrete instances of weakness as observed in real-world software or hardware products, typically denoted by CVE identifiers. These examples ground the abstract weakness in actual exploitation scenarios, aiding both validation and educational analysis.
- **relatedWeaknesses:** Links to other `CWE` instances that have hierarchical or peer-level relationships. Relations such as `ChildOf`, `ParentOf`, and `MemberOf` capture differing levels of abstraction, while `PeerOf` and `CanAlsoBe` denote lateral associations between weaknesses that share similar characteristics. This network of relations supports navigation across the weakness taxonomy and the discovery of alternate or sibling flaw patterns.
- **relatedAttackPatterns:** Connections to `AttackPattern` instances in the ontology, indicating adversarial techniques that are known to exploit the given weakness. By linking weaknesses to attack patterns, the model enables automated reasoning about likely exploitation vectors and the potential chaining of attack steps.

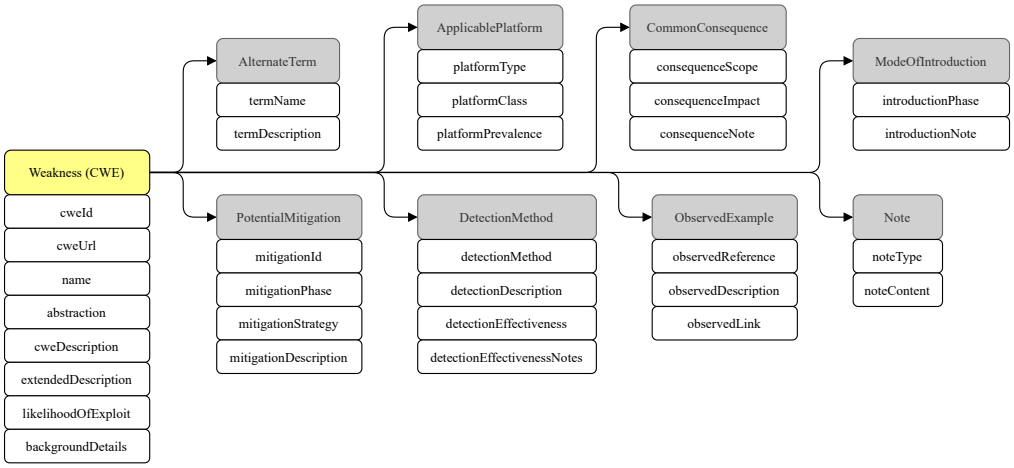

Figure 3: Class diagram illustrating the core CWE entity and its associated components in the proposed vulnerability ontology model.

3.3. CAPEC Class

The CAPEC class represents attack patterns that describe common techniques used by adversaries to exploit weaknesses in software and systems. As in [14], our ontological model extends the framework by formally modeling CAPEC attack patterns as a distinct class within the vulnerability ontology. This extension provides a comprehensive data schema for CAPEC instances, incorporating all information fields that can be systematically extracted from the official MITRE CAPEC data sources. While the original framework in [14] established the hierarchical structure of CAPEC data (categories containing meta-patterns, with each meta-pattern having multiple detail and standard patterns), our ontological modeling reveals additional complexity beyond the five core properties initially identified by the authors. Our analysis of actual MITRE data demonstrates that standard CAPEC entries contain not only the properties defined at the meta-level but also additional attributes specific to the standard abstraction level. Furthermore, some standard CAPEC patterns can contain detail-level CAPEC patterns within their structure, creating a more complex relational hierarchy than previously captured in the literature.

To address this complexity, we have incorporated the CAPEC `relatedTo` property into our ontological model, enabling the representation of hierarchical relationships between CAPEC instances across different abstraction levels (standard, detail, category). This property captures the parent-child relationships explicitly defined in MITRE's official data sources, where each CAPEC entry can be verified as either a `ChildOf` or `ParentOf` relationship with other CAPEC patterns. To distinguish between different types of CAPEC entries within our ontological framework, we utilize the `abstraction` field present in the source data, which explicitly defines whether a given CAPEC instance represents a category, meta-pattern, standard attack pattern, or detailed attack pattern. A figure of all the attribute we have considered for the Attack Pattern class is shown in Figure 4. We highlight five principal properties:

- **description:** A concise statement that characterizes the attack pattern, specifying the context in which the attack occurs, the actor's objectives, and the anticipated impact on the targeted systems. This field situates each CAPEC entry within real-world threat scenarios and clarifies its purpose and scope.
- **consequences:** Analogous to the `commonConsequences` in the CWE class, this structured list enumerates the negative outcomes that may result from the execution of the attack pattern. Each consequence entry identifies the violated security scope (e.g. confidentiality, integrity, availability), optionally describes the technical impact, and may include an ordinal probability of prioritize the most critical outcomes.
- **executionFlow:** A detailed, step-by-step account of the adversary's typical sequence of actions

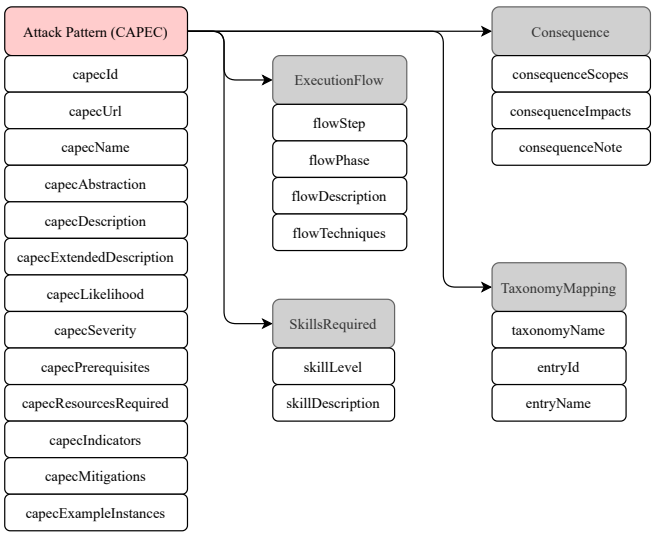

Figure 4: Class diagram illustrating the core CAPEC entity and its associated components in the proposed vulnerability ontology model.

when performing the attack. Organized into phases, such as *Explore*, *Experiment*, and *Exploit*, the execution flow delineates the preparatory reconnaissance, trial-and-error probing, and the final exploitation steps, thereby offering an operational view of the threat.

- **relatedWeaknesses:** References to one or more CWE instances whose presence is a precondition for the attack pattern's success. The association implies that any one of the listed weaknesses (though not necessarily all) must exist in the target environment to facilitate the adversary's technique. This property enables automated linkage from attack patterns back to the underlying technical flaws.
- **relatedAttackPatterns:** A network of lateral and hierarchical relationships to other CAPEC entries. Each `Related Attack Pattern` element includes an identifier and a `Nature` attribute that specifies the type of relation:
  - `ChildOf` and `ParentOf` denote abstraction levels, indicating broader or more specialized techniques;
  - `CanPrecede` and `CanFollow` capture sequential chaining in multi-stage attacks;
  - `CanAlsoBe` identifies patterns that, in certain contexts, may be interpreted as the target technique (note that this relation is not necessarily reciprocal);
  - `PeerOf` highlights analogous techniques that share similar objectives but do not fit other relational categories.

3.4. CPE Class and CWE Category Class

A fundamental distinction between our enhanced ontological model and the framework presented in [14] lies in the explicit modeling of two critical classes that were absent from the original work: the CPE (Common Platform Enumeration) class and the `CWECategory` class. These additions represent a significant advancement in achieving better coherence with the actual data structures present in authoritative vulnerability databases, moving beyond the simplified representations proposed in prior research.

3.4.1. CPE Class Enhancement

The CPE class represents software and hardware platforms that may be affected by vulnerabilities. Unlike [14], where platform information was treated as simple textual attribute, our ontological model elevates

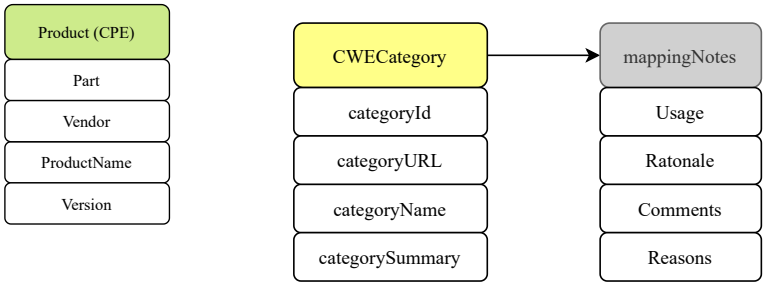

Figure 5: Class diagram illustrating the core CPE (on the left) and CWE Category (on the right) entities and its associated components in the proposed vulnerability ontology model.

CPE to a first-class entity with dedicated properties and relationships. Our data-driven analysis of official CVE databases reveals that CPE entries follow a structured format encompassing multiple semantic components (part, vendor, product, version, update, edition, language, software edition, target software, target hardware). To maintain ontological clarity while preserving essential semantic information, we have selected four core attributes that provide maximum value for vulnerability analysis:

- **part:** Distinguishes between applications, operating systems, and hardware devices
- **vendor:** Identifies the responsible organization or manufacturer
- **product:** Specifies the exact software application or hardware product
- **version:** Indicates the particular release or version number

3.4.2. CWE Category Class Introduction

The `CWECategory` class represents an entirely novel contribution absent from [14], addressing the hierarchical organization of weaknesses within the CWE taxonomy. Our enhanced model recognizes that MITRE organizes weaknesses into conceptual categories that group related security concerns. Our `CWECategory` class captures three fundamental properties derived directly from MITRE's official data structure:

- **categoryId:** The unique identifier assigned by MITRE to distinguish different weakness categories
- **categoryName:** The human-readable title that describes the category's focus area
- **categorySummary:** A comprehensive description of the types of weaknesses contained within the category

This categorical modeling enables the representation of `MemberOf` relationships between individual CWE instances and their parent categories, facilitating automated reasoning about weakness taxonomies and supporting more sophisticated vulnerability analysis workflows. The absence of such categorical representation in [14] limited the framework's ability to capture the full semantic richness of the CWE knowledge base, a gap that our enhanced ontological model explicitly addresses.

3.5. Data-integration Pipeline

To populate our ontology with real-world data, we developed a comprehensive pipeline using the Scrapy [20] framework for scalable, asynchronous web crawling and parsing. The process is shown in Figure 6.

The pipeline comprises three main stages:

First, in the *CVE Harvesting* stage, our custom Scrapy spider targets the NIST CVE API endpoint. Using Scrapy's built-in concurrency, the spider issues multiple parallel `GET` requests to retrieve the full set of CVE entries of interest. Upon receiving each JSON response, we extract core fields such as the

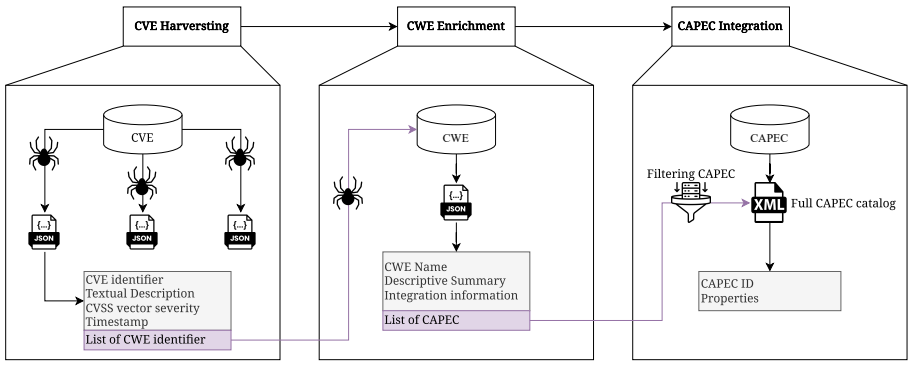

Figure 6: Description of the process of data integration for ontology population

CVE identifier, textual description, CVSS severity vector, and publication timestamp, as well as the list of associated CWE identifiers embedded within the CVE metadata.

Second, stage *CWE Enrichment* enriches each harvested CVE record with detailed weakness information. For every CWE ID collected in the previous step, the spider issues an on-the-fly GET request to the MITRE CWE API. This request returns structured details for the weakness, including its official name, descriptive summary, how such a weakness can be introduced and mitigated, and to what CAPEC it's exploited by. By performing this enrichment inline, rather than maintaining a separate crawl or static dump of CWE data, we ensure tight coupling between CVE and CWE records and reduce potential synchronization issues.

Finally, the *CAPEC Integration* stage handles the incorporation of the attack pattern data. Because MITRE does not provide a dedicated API for CAPEC, we downloaded the complete CAPEC catalog in XML format from the official repository. Given the modest size of the catalog relative to the CVE and CWE datasets, we opted for a single-pass batch parsing approach: the XML file is parsed in one go to extract each CAPEC ID and other properties, which then overwrite the placeholder ID from the CWE Enrichment stage.

Throughout the crawling and parsing process, we restricted all relationships to first-level links: CVE to CWE, CWE to CWE, etc., since our primary focus is on capturing the direct associations that underlie vulnerability exploitation. This streamlined pipeline produces a consistently structured high quality dataset ready for ingestion into the ontological model.

3.5.1. Database population

To validate the effectiveness and practical applicability of our enhanced ontological model, we designed and implemented a comprehensive empirical evaluation framework. This validation approach involves constructing a real-world vulnerability database populated according to our ontological schema, followed by systematic testing of the model's inference capabilities using actual vulnerability data.

For this purpose, we selected ArangoDB [21], a multi-model database that natively supports both graph and document data structures. This choice was specifically motivated by the hybrid nature of our ontological model, which requires efficient storage and querying of both rich entity attributes (document-oriented) and complex inter-entity relationships (graph-oriented). The multi-model approach enables us to preserve the complete semantic richness of vulnerability data while supporting sophisticated ontological reasoning operations. Following the data integration pipeline described above, the structured vulnerability dataset was systematically ingested into ArangoDB. The five core entity types—CVE, CWE, CWE Category, CAPEC, and CPE—were implemented as vertex collections, preserving their rich JSON document structure while enabling graph-based operations. The ontological relationships were materialized as edge collections, including causedBy, exploitedBy, affects, hasCategory and relatedTo connections. A subsection view of the graph database is shown in Figure 7.

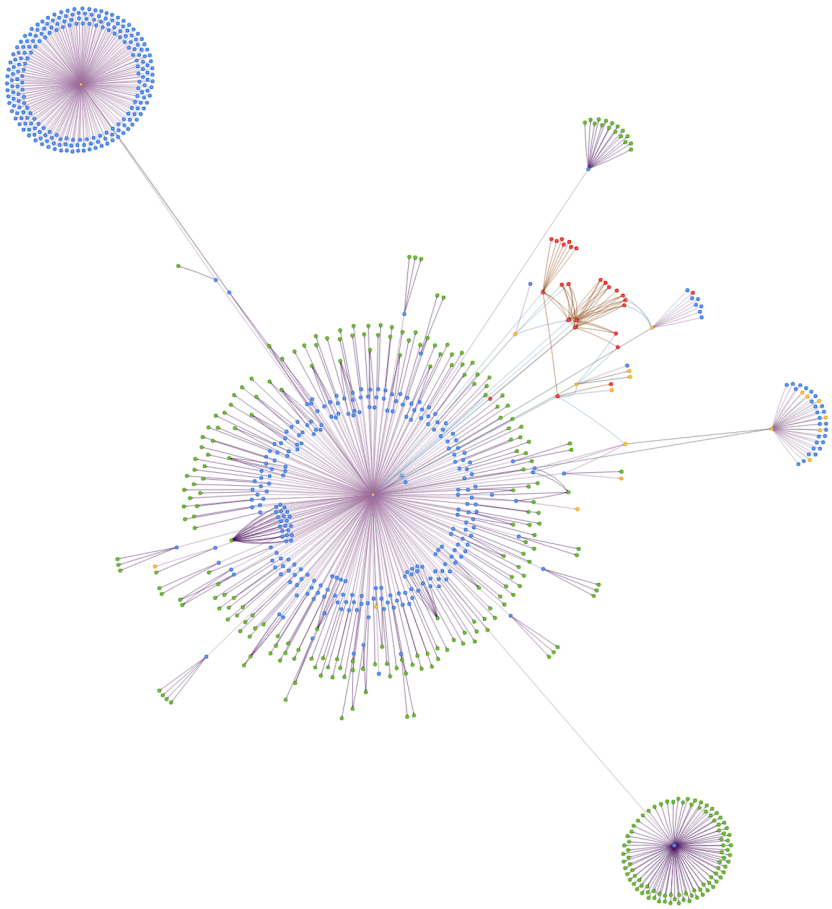

Figure 7: A subsection view of the Arangodb database, with nodes color-coded by document class (CVE in blue, CWE and CWE category in yellow, CAPEC in red and CPE in green)

3.6. Inference Rule

In order to demonstrate the effectiveness and flexibility of our model and its suitability for the data structure, we have developed an inference rule in SWRL (Semantic Web Rule Language) [22] that are specifically adapted to the structure of our dataset. This rule leverages the existing relationships between CVE, CWE, and CAPEC to infer new knowledge about the security landscape, enhancing the analytical capabilities of our vulnerability ontology framework [23]. Table 1 presents the pattern terms used in our inference rule, along with their definitions.

Table 1
Pattern Terms and Definitions

Pattern Term	Definition
CVE(?v)	Identifies a vulnerability instance ?v in the CVE collection
CWE(?w)	Identifies a weakness instance ?w in the CWE collection
CAPEC(?ap)	Identifies an attack pattern instance ?ap in the CAPEC collection
CPE(?c)	Identifies a platform/product instance ?c in the CPE collection
CAUSED_BY(?v, ?w)	Indicates that vulnerability ?v is caused by weakness ?w
EXPLOITED_BY(?w, ?ap)	Indicates that weakness ?w can be exploited by attack pattern ?ap
RELATED_TO(?w1, ?w2)	Indicates a relationship between weakness ?w1 and weakness ?w2
nature(?rt, ?n)	Specifies the nature ?n of relationship ?rt (e.g., "ChildOf")
AFFECTS(?v, ?c)	Indicates that vulnerability ?v affects platform/product ?c
potentialAttackChain(?v1, ?v2)	Inferred relationship indicating a potential attack progression from vulnerability ?v1 to ?v2

3.6.1. Attack Chain Inference Rule

We decided to implement an inference rule that identifies potential attack chains by connecting vulnerabilities that share related weaknesses and affect the same platform or product, helping to uncover multi-stage attack scenarios.

$$\begin{aligned} &\text{CVE}(?v1) \land \text{CVE}(?v2) \land \text{CWE}(?w1) \land \text{CWE}(?w2) \land \\ &\text{CAUSED\_BY}(?v1, ?w1) \land \text{CAUSED\_BY}(?v2, ?w2) \land \\ &\text{RELATED\_TO}(?w1, ?w2) \land \text{nature}(?rt, \text{"ChildOf"}) \land \\ &\text{CPE}(?c) \land \text{AFFECTS}(?v1, ?c) \land \text{AFFECTS}(?v2, ?c) \\ &\rightarrow \text{potentialAttackChain}(?v1, ?v2) \end{aligned} \tag{1}$$

To assess the effectiveness of the proposed inference rule, we implemented a database query using AQL (Arango Query Language) [21], translated from the inference rule we have defined. The following code shows the query used, and a sample of the results obtained is shown in Table 2.

```
FOR v1 IN CVE
  FILTER v1._key == "2022-40521"
  FOR v2 IN CVE
    FILTER v2._key == "2016-10408"
    FOR w1 IN CWE
      FOR cb1 IN CAUSED_BY
        FILTER cb1._from == v1._id AND cb1._to == w1._id
        FOR w2 IN CWE
          FOR cb2 IN CAUSED_BY
            FILTER cb2._from == v2._id AND cb2._to == w2._id
            FOR rt IN RELATED_TO
              FILTER rt._from == w1._id AND rt._to == w2._id AND rt.nature == "ChildOf"
              FOR c IN CPE
                FOR a1 IN AFFECTS
                  FILTER a1._from == v1._id AND a1._to == c._id
                  FOR a2 IN AFFECTS
                    FILTER a2._from == v2._id AND a2._to == c._id
                    RETURN {
                      "cve1": v1._key,
                      "cve2": v2._key,
                      "parent_cwe": w1._key,
                      "child_cwe": w2._key,
                      "affected_cpe": c._key,
                      "potentialAttackChain": CONCAT(v1._key, " -> ", v2._key)
                    }
```

Table 2
Results of the AQL query on the database

cve1	cve2	parent_cwe	child_cwe	affected_cpe	potentialAttackChain
2022-40521	2016-10408	287	284	o:qualcomm:apq8037_firmware:-	2022-40521 ->2016-10408
2022-40521	2016-10408	287	284	o:qualcomm:9206_lte_modem_firmware:-	2022-40521 ->2016-10408
2022-40521	2016-10408	287	284	o:qualcomm:sd820_firmware:-	2022-40521 ->2016-10408
2022-40521	2016-10408	287	284	o:qualcomm:sd626_firmware:-	2022-40521 ->2016-10408

The results in Table 2 demonstrate a concrete attack chain identified by our inference rule involving CVE-2022-40521 and CVE-2016-10408, both affecting multiple Qualcomm firmware platforms. CVE-2022-40521 is caused by CWE-287 (Improper Authentication), while CVE-2016-10408 is caused by CWE-284 (Improper Access Control). The inference rule successfully identified this as a potential attack chain because CWE-284 is a child weakness of CWE-287 in the MITRE taxonomy, representing a natural escalation path where an attacker could exploit authentication bypass vulnerabilities to gain

unauthorized access control.
This specific example illustrates a temporal attack chain spanning six years (2016-2022) across multiple Qualcomm firmware variants, including APQ8037, 9206 LTE modem, SD820, and SD626 platforms. The persistence of related weaknesses across different firmware generations creates a compound security risk where an attacker familiar with the older access control vulnerability (CVE-2016-10408) could leverage similar authentication bypass techniques against newer firmware (CVE-2022-40521). This finding highlights how firmware security debt accumulates over time, as legacy vulnerability patterns can inform exploitation strategies against newer systems that inherit similar architectural weaknesses. The practical significance of this discovered chain lies in its cross-platform nature across Qualcomm's product ecosystem. Security analysts managing Qualcomm-based systems should consider these vulnerabilities as interconnected threats rather than isolated incidents, particularly when multiple firmware versions coexist in the same organizational environment. The ability to automatically identify such temporal and cross-platform attack chains enables more comprehensive vulnerability assessment and prioritized remediation strategies that account for the compound risk of related weaknesses persisting across product generations.

4. Conclusion

In this paper we presented an enhanced data-driven ontology framework that systematically improves upon existing theoretical models for vulnerability knowledge representation. Our key contribution lies in addressing the critical gap between theoretical ontological frameworks and real-world vulnerability database structures through a comprehensive analysis of actual CVE, CWE, CAPEC, and CPE data sources. The enhanced ontological model introduces several important improvements over prior work: (1) the addition of classes such as CWE Category and CPE that are essential for complete data representation, (2) refinement of existing entity relationships to match actual database schemas, and (3) consolidation of fragmented properties into coherent, semantically meaningful attributes. These enhancements ensure full coherence between theoretical models and authoritative data sources. We validated our framework through practical implementation using a multi-model NoSQL database populated with real vulnerability data and demonstrated its effectiveness through SWRL-based inference rules. The attack chain inference rule successfully identified complex temporal relationships, illustrating how our framework enables automated discovery of compound security risks that would be difficult to detect through manual analysis.
The resulting knowledge base provides a foundation for advanced cybersecurity analysis, supporting both graph-based querying and automated reasoning about attack escalation paths. Future work will leverage this structured knowledge base to support natural language interfaces powered by Large Language Models [24, 25], enabling intuitive query capabilities and dynamic knowledge extraction across vulnerability relationships.

Acknowledgments

We acknowledge financial support from the PNRR MUR project PE0000013-FAIR.

Declaration of Generative AI

The author(s) have not employed any Generative AI tools

References

[1] D. W. Baker, S. M. Christey, W. H. Hill, D. E. Mann, The development of a common enumeration of vulnerabilities and exposures, in: Recent advances in intrusion detection, volume 7, 1999, p. 9.

[2] C. Vulnerabilities, Common vulnerabilities and exposures, The MITRE Corporation,[online] Available: https://cve. mitre. org/index. html (2005).
[3] S. Christey, J. Kenderdine, J. Mazella, B. Miles, Common weakness enumeration, Mitre Corporation (2013).
[4] B. A. Cheikes, D. Waltermire, K. Scarfone, Common platform enumeration: Naming specification version 2.3, NIST Interagency Report 7695 (2011).
[5] S. Barnum, Common attack pattern enumeration and classification (capec) schema, Department of Homeland Security (2008).
[6] H. Booth, D. Rike, G. A. Witte, The national vulnerability database (nvd): Overview (2013).
[7] U. S. S. of Documents, National Institute of Standards and Technology, US Government Printing Office, 1992.
[8] P. Mell, K. Scarfone, S. Romanosky, Common vulnerability scoring system, IEEE Security & Privacy 4 (2006) 85–89.
[9] S. A. Atiiq, C. Gehrmann, K. Dahlén, K. Khalil, From generalist to specialist: Exploring cwe-specific vulnerability detection, arXiv preprint arXiv:2408.02329 (2024).
[10] Y. Jiang, M. Jeusfeld, J. Ding, Evaluating the data inconsistency of open-source vulnerability repositories, in: Proceedings of the 16th International Conference on Availability, Reliability and Security, 2021, pp. 1–10.
[11] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, et al., Extensible markup language (xml) 1.0, 1998.
[12] M. A. I. Mallick, R. Nath, Navigating the cyber security landscape: A comprehensive review of cyber-attacks, emerging trends, and recent developments, World Scientific News 190 (2024) 1–69.
[13] T. UcedaVelez, M. M. Morana, Risk Centric Threat Modeling: process for attack simulation and threat analysis, John Wiley & Sons, 2015.
[14] L. Zhu, Z. Zhang, G. Xia, C. Jiang, Research on vulnerability ontology model, in: 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), 2019, pp. 657–661. doi:10.1109/ITAIC.2019.8785783.
[15] J. A. Wang, M. Guo, Ovm: an ontology for vulnerability management, CSIIRW '09, Association for Computing Machinery, New York, NY, USA, 2009. doi:10.1145/1558607.1558646.
[16] R. Syed, H. Zhong, Cybersecurity vulnerability management: An ontology-based conceptual model (2018).
[17] A. Ekelhart, S. Fenz, T. Neubauer, Ontology-based decision support for information security risk management, in: 2009 Fourth International Conference on Systems, IEEE, 2009, pp. 80–85.
[18] A. M. Rinaldi, C. Russo, C. Tommasino, Web document categorization using knowledge graph and semantic textual topic detection, in: Computational Science and Its Applications – ICCSA 2021, Springer International Publishing, 2021, pp. 40–51. doi:10.1007/978-3-030-86970-0_4.
[19] D. Benfenati, M. Montanaro, A. M. Rinaldi, C. Russo, C. Tommasino, Using focused crawlers with obfuscation techniques in the audio retrieval domain, in: Management of Digital EcoSystems, Springer Nature Switzerland, 2024, pp. 3–17. doi:10.1007/978-3-031-51643-6_1.
[20] D. Kouzis-Loukas, Learning scrapy, Packt Publishing Livery Place, 2016.
[21] ArangoDB, 2024. URL: https://arangodb.com/.
[22] A. SWRL, Semantic web rule language, 2013.
[23] C. Golbreich, Combining rule and ontology reasoners for the semantic web, in: International Workshop on Rules and Rule Markup Languages for the Semantic Web, Springer, 2004, pp. 6–22.
[24] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap, IEEE Transactions on Knowledge and Data Engineering 36 (2024) 3580–3599.
[25] D. Benfenati, G. M. De Filippis, A. M. Rinaldi, C. Russo, C. Tommasino, A retrieval-augmented generation application for question-answering in nutrigenetics domain, Procedia Computer Science 246 (2024) 586–595. URL: https://www.sciencedirect.com/science/article/pii/S1877050924025092. doi:https://doi.org/10.1016/j.procs.2024.09.467, 28th International Conference on Knowledge Based and Intelligent information and Engineering Systems (KES 2024).