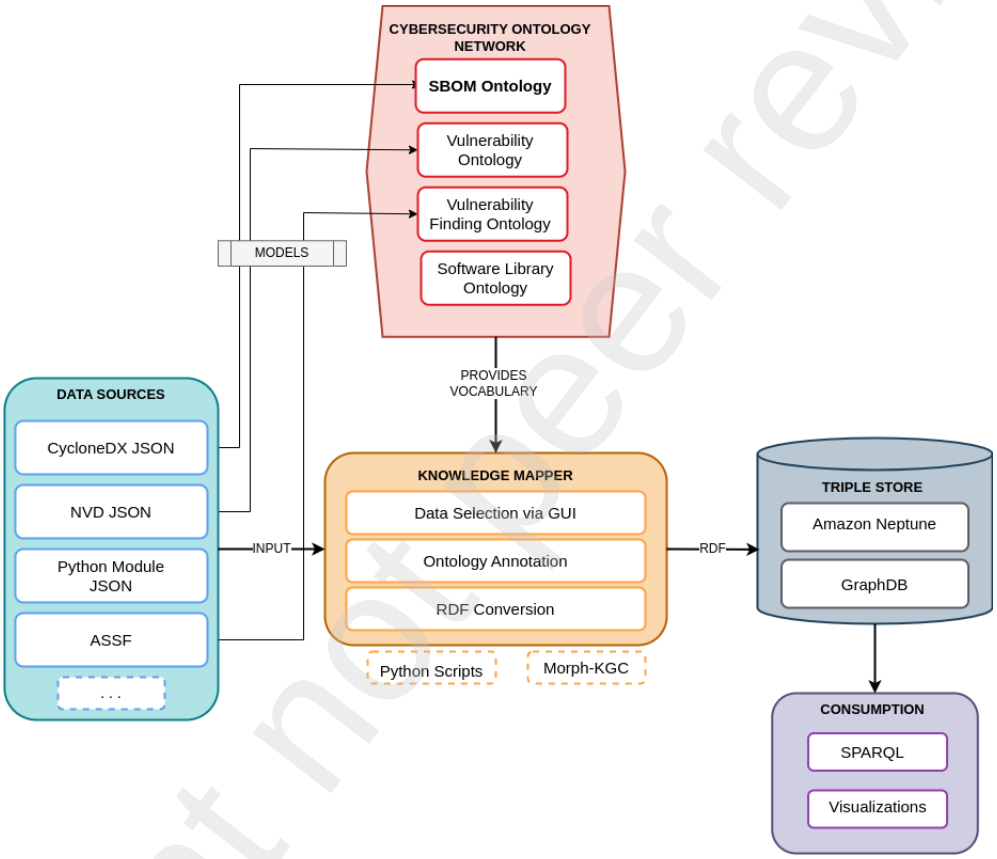
Graphical Abstract

A comprehensive view of software vulnerability risks through Enterprise Knowledge Graphs

Mikel Egaña Aranguren, Jesualdo Tomás Fernández-Breis, Bidane Leon Balentzia, Markus Rompe, Alexander García Castro



Highlights

A comprehensive view of software vulnerability risks through Enterprise Knowledge Graphs

Mikel Egaña Aranguren, Jesualdo Tomás Fernández-Breis, Bidane Leon Balentzia, Markus Rompe, Alexander García Castro

- Data model and ontology network to represent semantically software vulnerabilities
- Ontology modelling of the CycloneDX Bill of Materials Standard
- A proof-of-concept automated creation of a software vulnerability knowledge graph
- Knowledge graphs enhance cybersecurity analysis and decision-making

A comprehensive view of software vulnerability risks through Enterprise Knowledge Graphs

Mikel Egaña Aranguren[a], Jesualdo Tomás Fernández-Breis[b,*], Bidane Leon Balentzia[a], Markus Rompe[c], Alexander García Castro[c]

[a] *University of Basque Country (UPV/EHU), Bilbao, 48013, Basque Country, Spain*
[b] *Universidad de Murcia, CEIR Campus Mare Nostrum, IMIB-Pascual Parrilla, Murcia, 30100, Spain*
[c] *Siemens Energy, Germany*

Abstract

Cybersecurity has emerged as a critical concern for modern enterprises due to the increasing complexity and diversity of threats. These risks exploit multiple attack vectors, such as phishing, unpatched vulnerabilities, and malware distribution, necessitating a comprehensive and unified approach to threat modeling. However, cybersecurity data is often siloed across disparate sources—ranging from JSON vulnerability reports (e.g., Amazon Inspector, CycloneDX) and dependency files (e.g., NPM) to relational databases and manual assessments—making integration a significant challenge. Knowledge Graphs (KGs) offer the technological framework to successfully integrate disparate data. This work presents KG-based solution for software vulnerability data integration at Siemens Energy (SE), leveraging Enterprise Knowledge Graphs (EKGs) to unify heterogeneous datasets under a shared semantic model. Our approach consists of: (1) a Cybersecurity Ontology Network defining core entities and relationships, (2) an automated pipeline converting diverse data sources into a (3) scalable EKG that enables advanced threat analysis, and (4) competency questions validating the system's effectiveness. By adopting a Data-Centric Architecture, we demonstrate how EKGs provide a flexible, future-proof framework for cybersecurity intelligence, overcoming the limitations of traditional Application-Centric systems. This work offers actionable insights for organizations seeking to enhance cyber threat visibility while managing complex, evolving data landscapes.

*Corresponding author

Keywords: Cybersecurity, Knowledge Graph, ontology
PACS: 8920, 0705, 8805
2000 MSC: 6830, 6230

1. Introduction

Cybersecurity has become critical for most companies in the last years, as the nature of cybersecurity threats has evolved significantly in both complexity and impact [1, 2, 3, 4]. These threats now encompass numerous attack types and vectors; key attack types include ransomware attacks, Distributed Denial of Service, Man-in- the-Middle attacks, and Advanced Persistent Threats. These threats exploit various vectors, such as phishing campaigns, unpatched software vulnerabilities, malware distributions through jeopardized websites, and sophisticated supply chain attacks that compromise trusted distribution channels [5]. As organizations become more interconnected and reliant on ever-growing digital ecosystems, the diversity and subtlety of these attacks continue to expand. In order to obtain a useful and comprehensive view of the potential threats and vulnerabilities a company faces, data from diverse sources and processes needs to be integrated in a meaningful model. For example, software vulnerabilities might be detected in internal CI/CD pipelines by tools like Amazon Inspector [6], and stored in JSON, with a proprietary schema; other vulnerabilities might arise from other services, and stored in JSON, following the CycloneDX schema [7]; software dependencies might be stored in NPM JSON format; information about software owners, developers and project managers might be stored in a MySQL database or Excel spreadsheets; vulnerabilities might be manually detected, and their influence expanded to other software modules due to dependencies; and other unforeseeable sources of vital information.

Such data integration processes not only pertain to the cybersecurity domain: it is a general trend in current companies to move from an Application-Centric Architecture [8], in which information systems are designed around applications and services, to a Data-Centric Architecture [9], in which information systems are designed around data. This migration is complex and each company faces it differently, according to the available expertise and resources, technological debt, and culture. However, there is a clear trend towards the use of Knowledge Graphs (KGs) [10] as vehicles for enterprise data integration, in the form of Enterprise Knowledge Graphs (EKGs) [11]. EKGs

offer the necessary flexibility to accommodate heterogeneous data while preserving a common data model. Furthermore, since EKGs include carefully designed vocabularies (Ontologies), a *lingua franca* is implemented to describe and include new entities that might arise in future integration efforts, offering a robust, future-proof, and interoperable data infrastructure that complies with FAIR principles (Findable, Accessible, Interoperable, Reusable) [12].

In the last years, there has been an increasing interest in ontologies and knowledge graphs for cybersecurity. On the ontology side, a few ontologies have been proposed to address diverse needs, including threat intelligence, incident response, vulnerability management, and cyber-physical system security. For example, the Unified Cybersecurity Ontology (UCO) integrates existing standards such as STIX, CVE, CWE, and CAPEC, providing a comprehensive semantic layer for security information sharing and analysis [13]. The Internet of Things Security Ontology (IoTSec) [14] focuses on representing assets, vulnerabilities, and countermeasures specific to IoT ecosystems. The Reachability Matrix Ontology [15] describes and evaluates reachability and attack paths in network security contexts. A systematic review of cybersecurity ontologies highlights that while numerous ontologies exist for focused domains, constructing a truly generic and extensible cybersecurity ontology remains a challenge due to the evolving nature of the cyber threat landscape.

In this work, we are interested in contributing to the specific domain of software vulnerabilities. An initial attempt was carried out in [16] provided an ontology for vulnerability management, without paying attention to important aspects related to the software libraries and the software bill of materials. A more recent effort with similar limitations was presented in [17]. In [18] a model for software defects was provided but, again, it just focused on one aspect relevant for the effective management of software vulnerabilities.

On the cybersecurity knowledge graphs side, they have been applied to different tasks such as risk assessment [19],cyber-attack detection [20, 21], uncovering CWE-CVE-CPE relations [22], threat intelligence [23]. Large language models have also been applied in this area to generate cybersecurity knowledge graphs [24], or to evaluate vulnerabilities [25]. The importance of cybersecurity knowledge graphs has led to the development of methods for assessing the quality of cybersecurity KGs [26]. The review presented in [27] shows that the cybersecurity KGs have as main focus the detection of intrusions, risk assessment or malware detection, focusing on threats and attack patterns as research object. However the connections between software

packages, libraries and vulnerabilities have not been sufficiently addressed to date. [27] also identifies the challenge of exploiting log data for cybersecurity management.

In this work we describe the application of EKGs to integrate software vulnerability data at Siemens Energy (SE) [28], as a first step towards a FAIR and Data-Centric system for managing cybersecurity knowledge. The goal is to provide cybersecurity analysts with information to quickly identify which software packages and libraries may be affected by a vulnerability detected by state of the art security inspector tools, whose information is recorded in structured logs. Our approach comprises the following elements: the Cybersecurity Ontology Network, a vocabulary to represent common software vulnerability entities; the pipeline that, using the vocabulary, converts and integrates data from diverse sources into the EKG (Cybersecurity Knowledge Graph); the competency questions used to evaluate the system. These elements, in combination, provide a robust infrastructure to integrate and process cybersecurity related information to find meaningful facts. The architecture of this work is based on semantic technologies: RDF (Resource Description Framework) [29] for data codification, OWL (Web Ontology Language) [30] for Knowledge Representation, and SPARQL (SPARQL Protocol and RDF Query Language) [31] for querying. It should be noted that our ontology network will be driven by the CycloneDX standards for specifying software bills of materials, which have an increasing industry implantation.

In summary, this paper contributes to the field by introducing an ontology-driven, enterprise-scale knowledge graph architecture for software vulnerability management, demonstrating how CycloneDX SBOMs and related vulnerability data can be semantically integrated and queried to support threat modeling and operational security intelligence.

The rest of the paper is organised as follows: section 2 describes the implemented methods, including the data sources (Section 2.2), the developed ontologies (Section 2.1), the mappings from the data sources to the Knowledge Graph (Section 2.3), data storage (Section 2.4) and data consumption (Section 3.2); section 3 provides an overview of the resulting Knowledge Graph and it describes the competency questions used for the evaluation; section 4 discusses the problems and decisions taken during the conversion process; finally, section 5 provides the conclusions of the work and future directions.

2. Materials and methods

The general workflow for the generation of the Software Vulnerabilities Knowledge Graph can be seen in figure 1. The workflow starts in the data sources (Section 2.2) and converts them to RDF by executing the conversions defined in the Knowledge Mapper (Section 2.3), based on the vocabularies provided by the Cybersecurity Ontology Network (Section 2.1); then, the RDF produced as a result of the conversion is stored in the Triple Store (Section 2.4) for further consumption (Section 3.2).

2.1. The Cybersecurity Ontology Network

The Cybersecurity Ontology Network offers the vocabulary to describe the cybersecurity data of the domain. It comprises the following ontologies, linked at different key points to build a network.

Software Bill of Materials (SBOM) ontology. The Software Bill of Materials Ontology is the main element of the CyberSecurity Ontology Network, and it is used to semantically describe Software Bill of Materials files generated in CI pipelines that follow the CycloneDX specification. The upper classes of the ontology can be seen in Figure 2.

Vulnerability ontology. The Vulnerabity Ontology provides the terms and properties for describing the data about vulnerabilities that can affect software components included in the software projects in use in Siemens Energy, and that will be the vulnerabilities reported by AWS or other security analytics services (Figure 3). The scope of the ontology is to describe the information about vulnerabilities offered by standardization organizations, including the description of the vulnerablity, their impact and types of problems generated by the vulnerability. In this case, the scope is limited by the U.S. government repository of standards based vulnerability management model, that is, the specification of the National Vulnerabilty Database (NVD) of the US National Institute of Standards and Technology (NIST). The ontology has been developed to provide a data model based on the NVD JSON Schema 1.1. This schema uses other schemas for particular content: CVE vulnerabilities [32], CVSS V2.0 scores [33], and CVSS V3.0 scores [34]. The current version models all the elements included in those schemas.

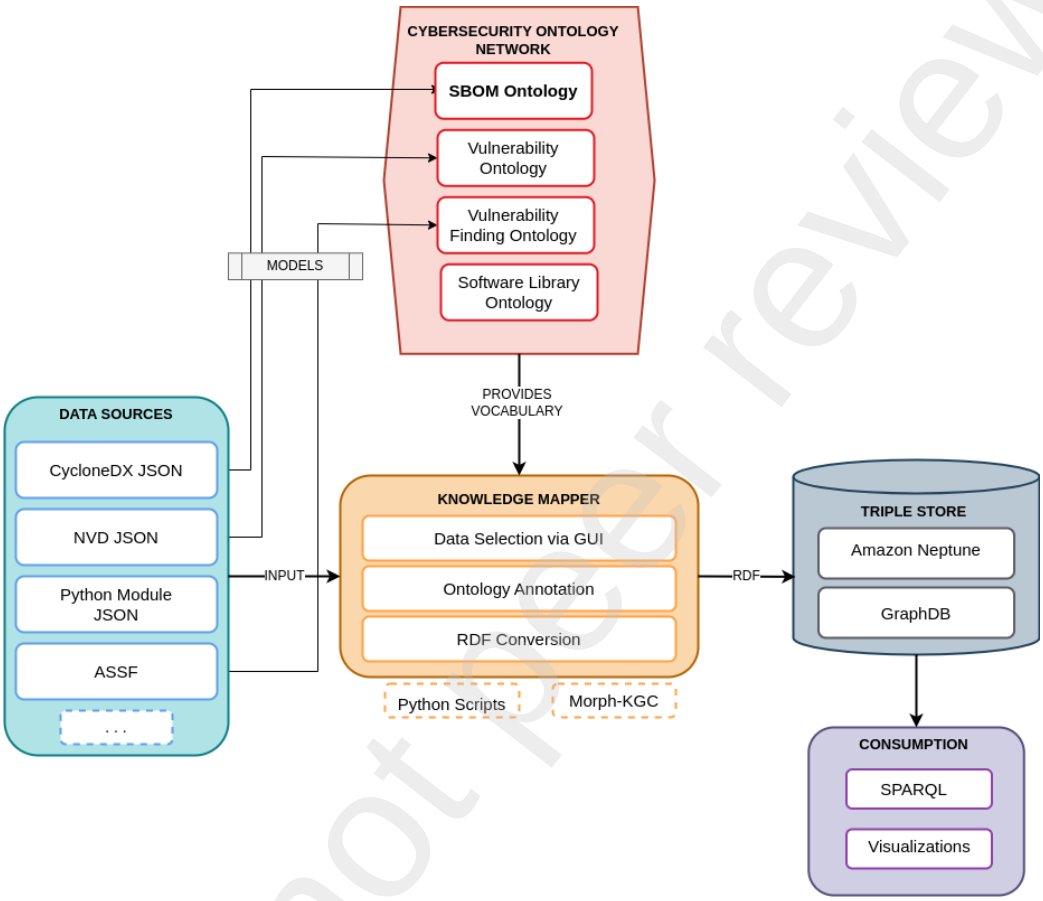


Figure 1: General Worfklow for the generation of the Cybersecurity Knowledge Graph. Data Sources (left, blue): Various formats such as CycloneDX JSON, NVD JSON, Python Module JSON, and ASSF provide input data; it is expected that more data sources will be added in the future. The Cybersecurity Ontology Network (top, red), comprising the SBOM Ontology, Vulnerability Ontology, Vulnerability Finding Ontology, and Software Library Ontology, defines the vocabulary for the domain. The Knowledge Mapper (center, orange) receives input data, allows data selection via a GUI, annotates it with ontology terms, and converts it to RDF (Python scripts or Morph-KGC are also used for the conversion). Triple Store (right, blue): The generated RDF is stored in graph databases such as Amazon Neptune. Consumption (bottom right, purple): The RDF data can be queried using SPARQL or visualized. Arrows indicate the flow of data and vocabulary between components. The ontologies provide the semantic model for annotation and mapping, enabling integration and querying of heterogeneous cybersecurity data.

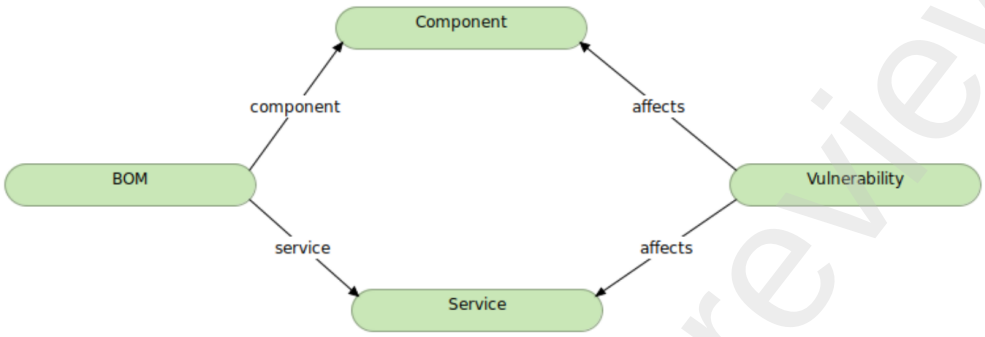


Figure 2: Software Bill of Materials ontology. A Bill Of Materials includes, amongst other metadata, components (e.g. software libraries) and services, that can be affected by vulnerabilities. Other classes are not included in the diagram.

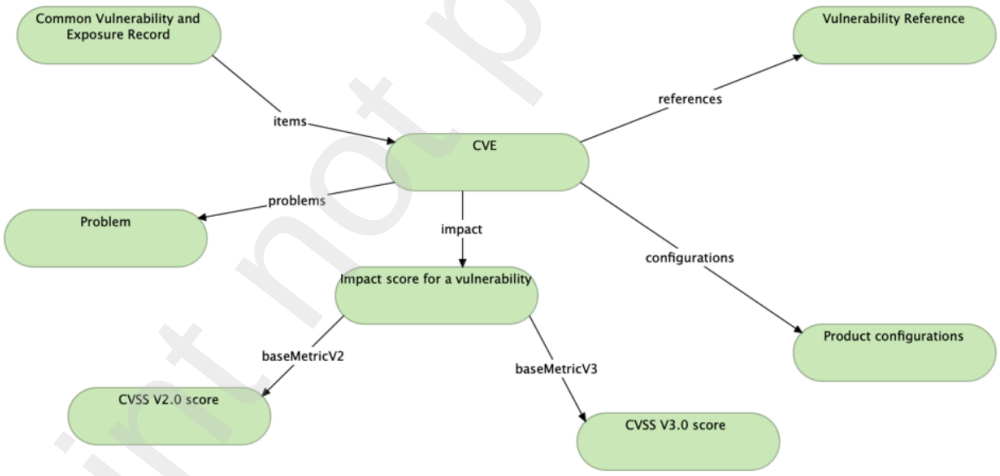


Figure 3: Vulnerability ontology. NVD describes the data in terms of record, and each record provides information about a set of Common Vulnerabilities and Exposures (CVE). Each CVE is described by metadata (not shown in the figure), the score of the impact of the vulnerability, the types of problems it may cause, references describing the vulnerability and additional information such as configurations.

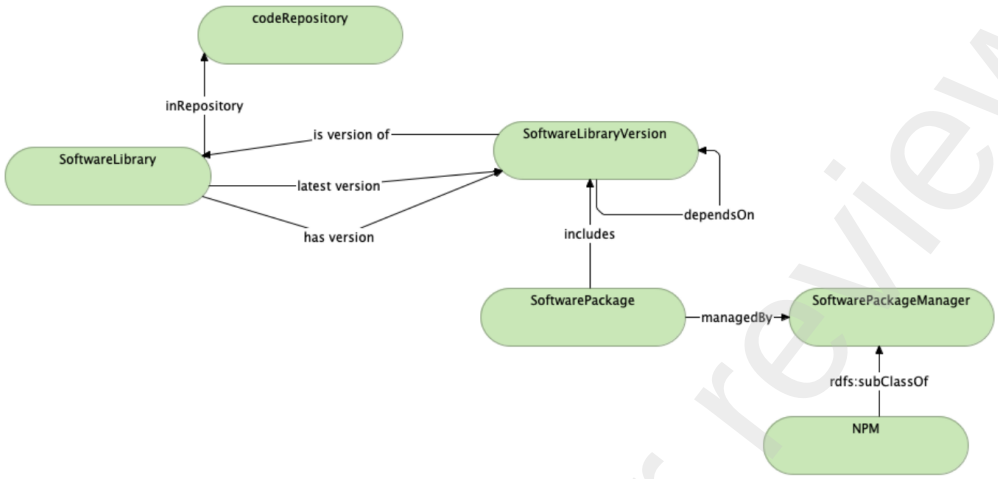


Figure 4: Software Library ontology. The software library and its versions are the main entities, which have several relations between them. The model also represents the fact that a software library version has dependencies with versions from different software libraries, that the software library is stored in a code repository and that the distribution of the library versions is through software package managers.

Software Library ontology. The scope of the ontology is to describe the information about software libraries generated by software package managers. It provides the terms and properties for describing the data about software libraries included in the software projects in use in Siemens Energy and that could be the subject of vulnerabilities reported by AWS or other security analytics services. This ontology is reused by the other two ontologies to represent the sofware components of the SBOM and the software libraries for which vulnerabilities have been identified. The upper classes of the ontology can be seen in Figure 4.

Vulnerability finding ontology. The scope of the ontology is to describe the information about vulnerability findings on software libraries generated by the AWS Security Hub [35]. The ontology has been developed to provide a data model based on JSON Schema of the AWS Security Finding Format (ASFF) [36]. The current version does not implement the whole format, only the fields included in the data samples provided by Siemens Energy, which is also delimiting the scope of this ontology. It provides the terms and properties for describing the findings about vulnerabilities in software libraries included in the software projects in use in Siemens Energy and deployed in AWS. The

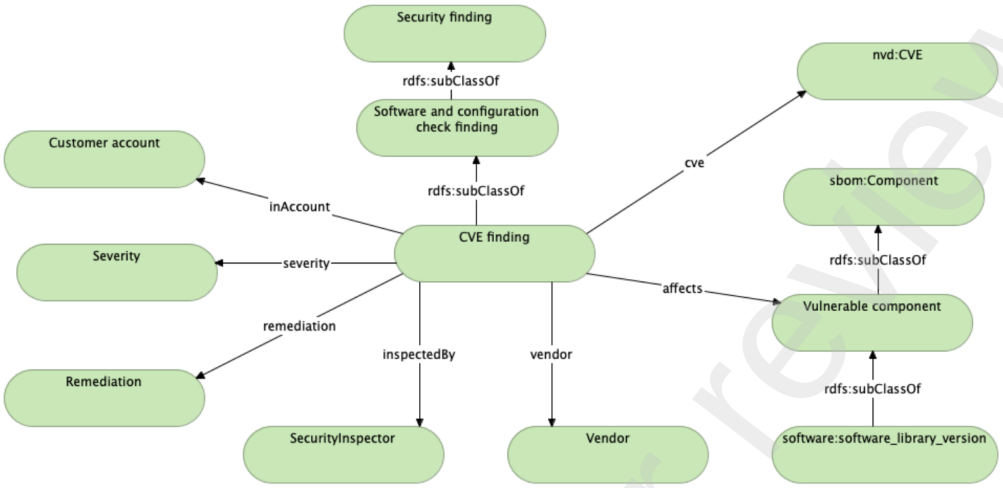


Figure 5: Vulnerability finding ontology. A CVE finding affects a software library version and is due to a common vulnerability and exposure (CVE). The finding is described by the severity of the vulnerability, the vendor, the security inspector and in which account (AWS in this case) it has been identified.

upper classes of the ontology can be seen in Figure 5.

2.2. Data sources

The cybersecurity information needed for a company to have a comprehensive view of the current problems is compiled by internal and external sources. The data gathered from internal sources is the result of analysis tools. External sources are public databases that contain additional information about the detected problems. The information is generally structured in JSON files, easing its parsing: however, the schemas of those files are designed for syntactic interoperability rather than semantic interoperability, and hecne the need for the vocabulary described in Section 2.1. The following data sources are the main ones used for the current implementation of the pipeline.

2.2.1. CycloneDX JSON files

CycloneDX Bill of Materials (BOM) represents a full-stack inventory of software, hardware, services and other assets, and serves as a specification for creating Software Bill of Materials (SBOM) files. An SBOM file is an structured description of components, libraries and modules required

to build, maintain and deploy software through the whole life cycle. CycloneDX has become the ECMA-424 standard [37]. An SBOM file typically includes a component Inventory (Lists all software components, including libraries, frameworks, and tools), a dependency graph (Relations between components), information about vulnerabilities affecting components, metadata (Details about the SBOM itself, such as version, author, and timestamp) and license Information for each component (Figure 6).

2.2.2. NVD JSON files

Vulnerabilities in Cybersecurity are normally represented as CVEs (Common Vulnerabilities and Exposures). The CVE system [38] is maintained by the MITRE Corporation [39] and provides a standardized way to identify and reference vulnerabilities across different tools, databases, and organizations. The National Vulnerability Database (NVD) [40] provides JSON feeds that contain detailed and enriched information about CVEs, including their descriptions, severity scores (CVSS), affected software, and references [41]. These JSON files are updated regularly and are widely used by security professionals, researchers, and tools to analyze and manage vulnerabilities. NVD JSON Files typically include the CVE unique ID (e.g., CVE-2023-1234), a description, CVSS Metrics (Common Vulnerability Scoring System -CVSS- scores and vectors to assess the severity), references (Links to advisories, patches, and additional resources), affected products (details about the software or hardware impacted by the vulnerability), and published and modified dates (timestamps for when the vulnerability was disclosed and last updated) (Figure 7).

2.2.3. Software registries

A software registry is a centralized repository or database where information about software components is stored, managed, and made accessible to users and systems. A software registry is a key part of the infrastructure for developing, distributing, and managing software, whether for code packages, containers, or system configurations. The specific meaning depends on context. In this work we have worked with data from Javascript and Python software libraries both. Therefore, the data of these libraries was provided as NPM and PyPI packages.

```
{
  "$schema": "http://cyclonedx.org/schema/bom-1.4.schema.json",
  "bomFormat": "CycloneDX",
  "specVersion": "1.4",
  "version": 1,
  "serialNumber": "urn:uuid:a4d3d687-b56f-4ec9-ae2a-daf44203e9c2",
  "metadata": {
    "timestamp": "2024-05-03T07:16:48.781Z",
    "tools": [
      {
        "name": "npm",
        "version": "9.5.1"
      },
      {
        "vendor": "@cyclonedx",
        "name": "cyclonedx-npm",
        "version": "1.17.0",
        "externalReferences": [
          {
            "url": "git+https://github.com/CycloneDX/cyclonedx-node-npm.git",
            "type": "vcs",
            "comment": "as detected from PackageJson property \"repository.url\""
          },
          {
            "url": "https://github.com/CycloneDX/cyclonedx-node-npm#readme",
            "type": "website",
            "comment": "as detected from PackageJson property \"homepage\""
          },
          {
            "url": "https://github.com/CycloneDX/cyclonedx-node-npm/issues",
            "type": "issue-tracker",
            "comment": "as detected from PackageJson property \"bugs.url\""
          }
        ]
      },
```


Figure 6: This is a partial view of a CycloneDX Software Bill of Materials (SBOM) in JSON format. It describes metadata about the SBOM, including the schema version, format, serial number, timestamp, and tools used to generate the SBOM (such as npm and CycloneDX tools). Each tool entry may include its name, version, vendor, and external references like repository URLs, websites, and issue trackers.

```
{
  "CVE_data_type" : "CVE",
  "CVE_data_format" : "MITRE",
  "CVE_data_version" : "4.0",
  "CVE_data_numberOfCVEs" : "2723",
  "CVE_data_timestamp" : "2025-02-14T08:00Z",
  "CVE_Items" : [ {
    "cve" : {
      "data_type" : "CVE",
      "data_format" : "MITRE",
      "data_version" : "4.0",
      "CVE_data_meta" : {
        "ID" : "CVE-2025-0015",
        "ASSIGNER" : "arm-security@arm.com"
      },
      "problemtype" : {
        "problemtype_data" : [ {
          "description" : [ ]
        } ]
      },
      "references" : {
        "reference_data" : [ {
          "url" : "https://developer.arm.com/Arm%20Security%20Center/Mali%20GPU%20Driver%20Vulnerabilities",
          "name" : "https://developer.arm.com/Arm%20Security%20Center/Mali%20GPU%20Driver%20Vulnerabilities",
          "refsource" : "",
          "tags" : [ ]
        } ]
      },
      "description" : {
        "description_data" : [ {
          "lang" : "en",
          "value" : "Use After Free vulnerability in Arm Ltd Valhall GPU Kernel Driver, Arm Ltd Arm 5th Gen GPU
          Architecture Kernel Driver allows a local non-privileged user process to make improper GPU processing
          operations to gain access to already freed memory.This issue affects Valhall GPU Kernel Driver: from
          r48p0 through r49p1, from r50p0 through r52p0; Arm 5th Gen GPU Architecture Kernel Driver: from r48p0
          through r49p1, from r50p0 through r52p0."
        } ]
      }
    },
```


Figure 7: This image shows a snippet of a JSON file containing CVE (Common Vulnerabilities and Exposures) data in the MITRE/NVD format. The file includes metadata about the CVE list and detailed entries for individual CVEs, such as CVE-2025-0015, with fields for assigner, problem type, references, and description. The structure is typical for NVD JSON feeds used in vulnerability management and cybersecurity automation.

```
{
    "Findings": [
        {
            "SchemaVersion": "2018-10-08",
            "Id": "arn:aws:inspector2:eu-central-1:818294812452:finding/fea996c8cff97564a365cd59d2c1e36a",
            "ProductArn": "arn:aws:securityhub:eu-central-1::product/aws/inspector",
            "ProductName": "Inspector",
            "CompanyName": "Amazon",
            "Region": "eu-central-1",
            "GeneratorId": "AWSInspector",
            "AwsAccountId": "818294812452",
            "Types": [
                "Software and Configuration Checks/Vulnerabilities/CVE"
            ],
            "FirstObservedAt": "2023-12-20T06:09:28Z",
            "LastObservedAt": "2023-12-20T06:09:28Z",
            "CreatedAt": "2023-12-20T06:09:28Z",
            "UpdatedAt": "2023-12-20T06:09:28Z",
            "Severity": {
                "Label": "MEDIUM",
                "Normalized": 40
            },
            "Title": "CVE-2020-7598 - minimist",
```


Figure 8: This image shows a JSON file containing a security finding from AWS Inspector. The finding includes metadata such as schema version, finding ID, product and company information, region, generator ID, account ID, finding types, timestamps, severity, and a CVE title. This format is typically used for reporting vulnerabilities and configuration issues in cloud environments.

2.2.4. AWS Security Finding Format (ASFF) files

Amazon Inspector is an automated vulnerability assessment service offered by Amazon Web Services (AWS) that enhances cloud security by continuously scanning EC2 instances for software vulnerabilities, network misconfigurations, and compliance deviations, leveraging CVE databases. Amazon Inspector produces findings in AWS Security Finding Format (ASFF), a JSON-based file format that includes, most importantly, found vulnerabilities, their CVSS score, and the affected software packages (Figure 8).

2.3. Mapping

The conversion process of source data (e.g. JSON files) to RDF is carried out using the Knowledge Mapper, a tool developed internally at SE (Figure 9). The Knowledge Mapper offers a Graphical User Interface (GUI) that allows the user to choose a data source, annotate it with ontologies, and convert it to RDF. The resulting RDF is stored directly in the Triple Store (Section 2.4). Additionally, the Knowledge Mapper saves the defined mapping in the YARRRML format [42], making the mapping reproducible at any time in the future.

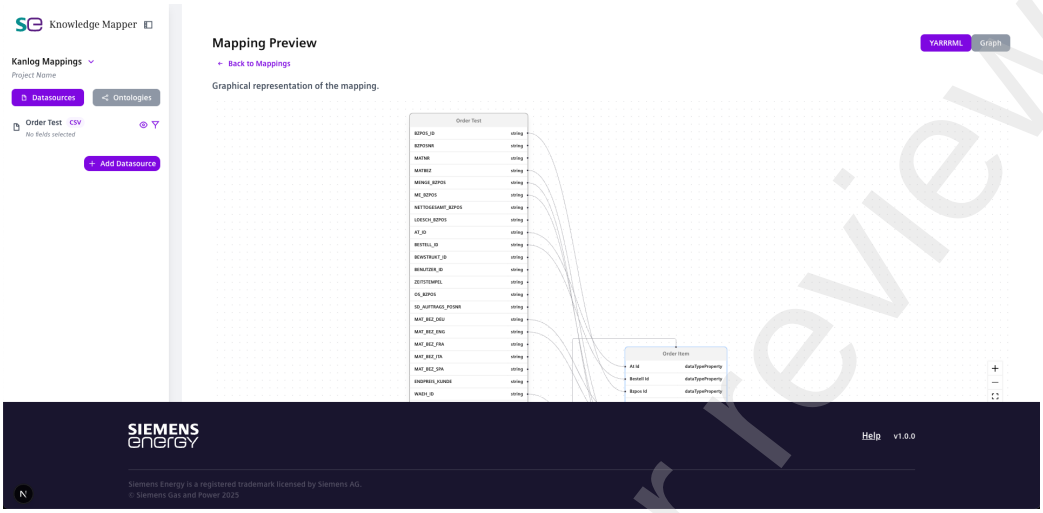


Figure 9: Knowledge Mapper interface. The interface shows a data sources pane on the left, and a main display on the center, in which elements from the data sources are connected to ontologies, to do the mapping. Once the mapping has been defined, it can be executed and the resulting RDF is stored in the Triple Store.

A conceptual representation of the mapping of data sources to RDF is shown in Figure 11. The mapping process generates an RDF Named Graph for each data source, containing, for example, the data from SBOM JSON files. However, the Named Graphs are not connected, and therefore a linking process is set up. The linking process detects package names that can be found in SBOM descriptions and AWS ASFF descriptions, therefore detecting SBOM libraries (Components) that have vulnerabilities detected by AWS ASFF (Figure 12). The linking process is executed by an SPARQL SELECT/INSERT query, and it uses Named Graphs to organize the discovered links (Figure 10).

2.4. Storage

Semantic technologies like RDF and (RDF-serialized) OWL offer the guarantee of loading into any W3C standard compliant Triple Store, enhancing technical interoperability.

Data is organized within the Triple Store as a collection of Named Graphs. A Named Graph is a collection of triples identified by a URI. That URI can be used as the subject for further triples, effectively using the same language (RDF) for data (Triples within the Named Graph) and metadata

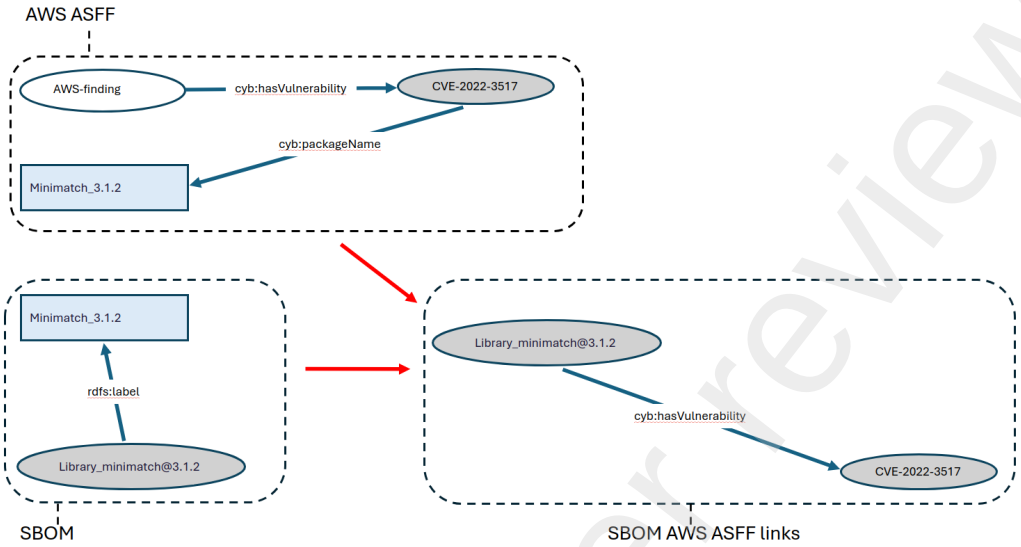
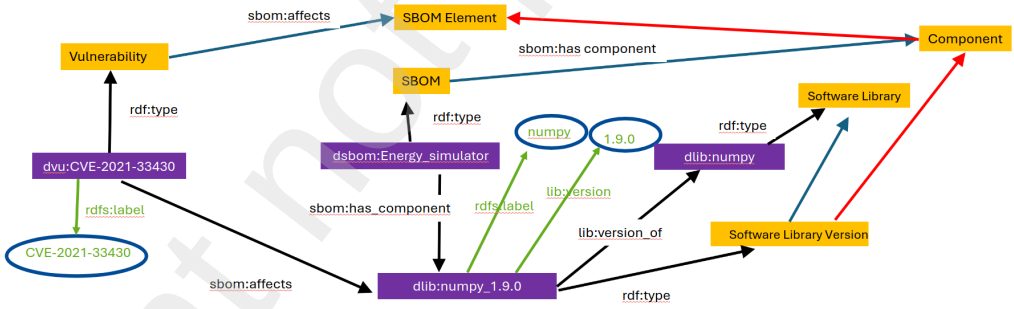


Figure 10: Linking process (Short names instead of URIs are used for Named Graphs, for the sake of clarity). The data to be linked is contained in two Named Graphs, AWS-ASFF and SBOM: both contain a triple that refers to the same name, Minimatch_3.1.2, through different predicates. The linking process detects the name and creates a further link in another Named Graph, SBOM-AWS-ASFF-links.



Project name	Component	name	version	CVE
Energy simulator	Library	numpy	1.9.0	CVE-2021-33430
CaPyCLI	Library	alembic	1.13.0	
CaPyCLI	Library	flask	2.3.1	CVE-2023-30861

Figure 11: Mapping example. The table bellow summarises the data from a sample JSON file. The Knowledge Graph on the top renders the data from the JSON file, including its annotation to the pertinent ontology classes. Blue arrow: object property; green arrow: data property; red arrow: subClassOf; black arrow: rdf:type.

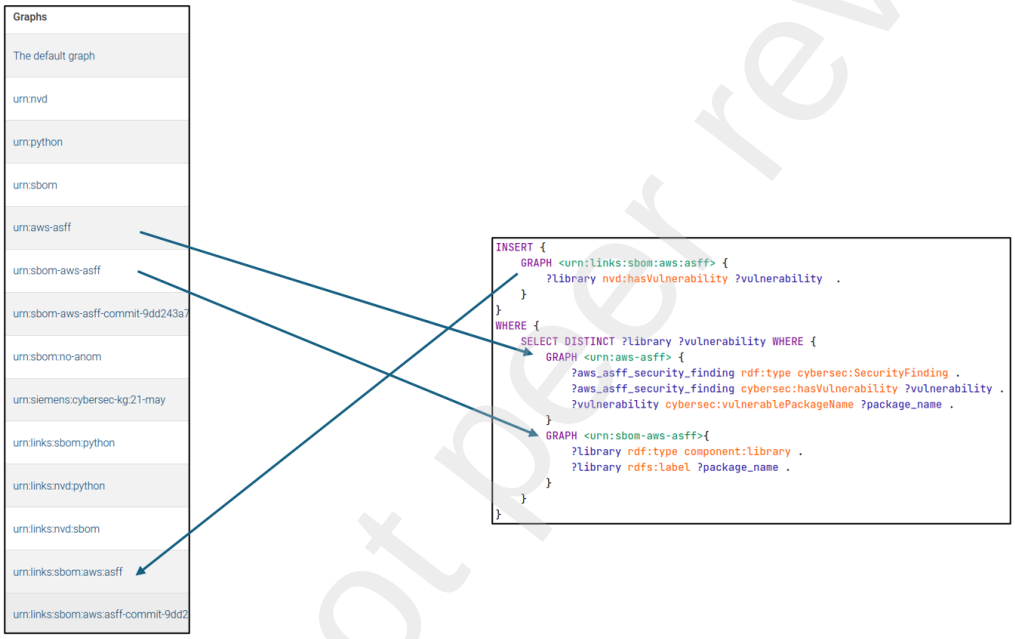


Figure 12: This figure illustrates the process of linking security findings from AWS ASFF data to software component libraries using SPARQL. The left panel lists available RDF Named Graphs, while the right panel shows a SPARQL INSERT query that creates the approrpiate relationships between libraries and vulnerabilities. This process effectivelly integrates data from the urn:aws-asff and urn:sbom-aws-asff Named Graphs, and writes the results to the urn:links:sbom:aws:asff Named Graph. Arrows indicate the flow of data between the relevant Named Graphs and the query logic.

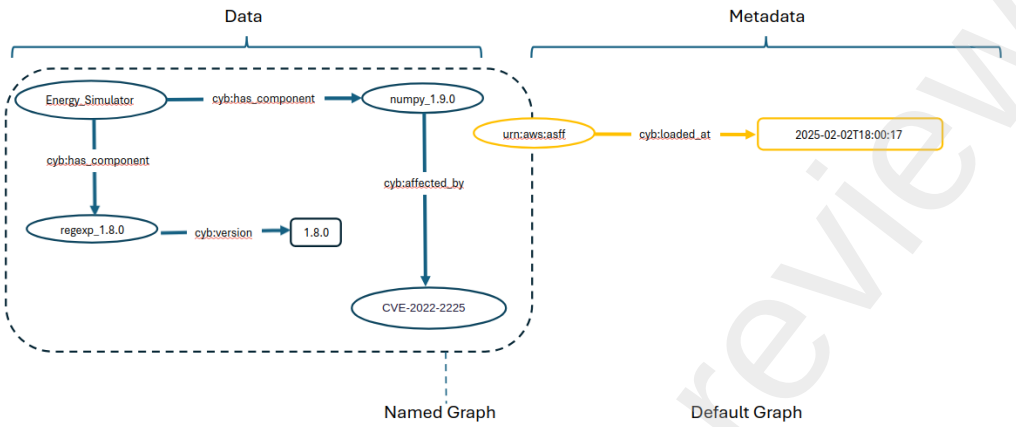


Figure 13: Named Graphs. Data (On the left) is stored in a Named Graph identified by a URI (Yellow circle in the center. Metadata referring to the Named Graph, on the right, is also codified in RDF and stored in the Default Graph (The graph storing all the triples of the repository).

(Triples in the Default Graph referring to the Named Graph URI). This mechanism offers an advantage over the relational model, making information management more flexible and efficient, for example for provenance recording (Figure 13).

3. Results

3.1. An Enterprise Knowledge Graph for software vulnerabilities information

The result of executing the pipeline described in Section 2 is a collection of RDF Named Graphs that integrate the cybersecurity-related information from data sources. The RDF data is also annotated with entities from the Cybersecurity Ontology Network, hence offering a *Knowledge* Graph. Provenance and metadata are also collected for each Named Graph through the appropriate vocabularies like DCAT [43] and PROV [44].

3.2. Graph consumption

Once the data has been converted to RDF and stored in the triple store, it can be consumed by the human users (SE data analysts) and/or other agents like applications for Business Intelligence (e.g. dashboards). There are currently three modes for exploration of the data:

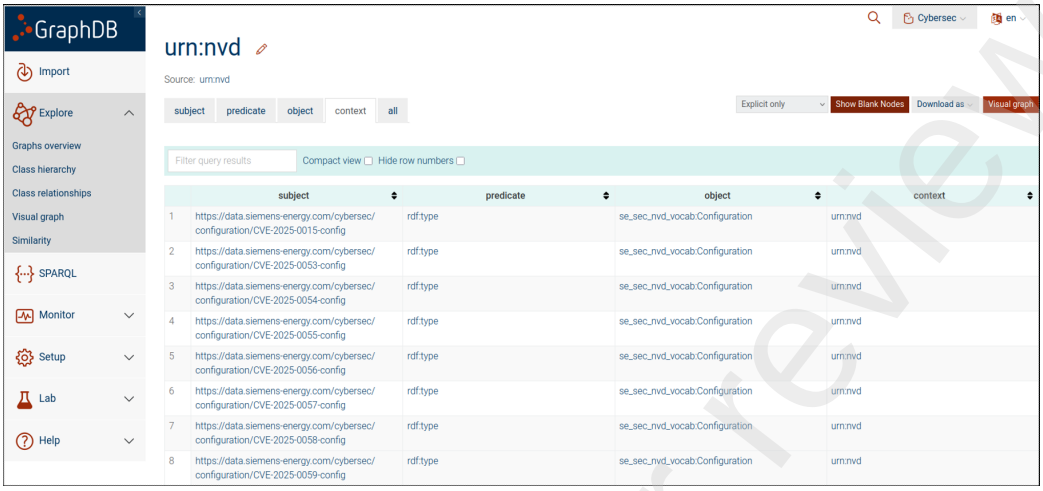


Figure 14: This figure displays the GraphDB Explore interface for the urn:nvd named graph, presenting a tabular view of RDF triples. Each row lists a subject, predicate, and object, representing configuration entities for various CVEs as typed by the NVD vocabulary, facilitating exploration of vulnerability metadata.

Triples exploration (Figure 14). This interface shows the RDF triples in tabular form. This interface is for technical developers to explore specific areas of the graphs.

SPARQL interface (Figure 15). The SPARQL interface allows users to pose advanced SPARQL queries against the data, obtaining the results in the appropriate format. More importantly, the SPARQL endpoint can be accessed by any agent through the SPARQL protocol [45].

Graph visualisation (Figure 16). The Graph visualisation allows end users to explore the graph visually.

3.3. Evaluation

The resulting EKG was evaluated by performing a set of predefined competency questions. Those competency questions were gathered from SE staff related to the cybersecurity domain, and translated to SPARQL by developers. A sample of the competency questions, their translation to SPARQL and the corresponding answer are shown in Listings 1, 2 and Tables 2, 2.

Figure 15: This figure shows the GraphDB SPARQL Query and Update interface, where a SPARQL query is constructed to correlate CVE vulnerability descriptions from the NVD dataset with software dependencies extracted from a Python SBOM. The query uses multiple RDF prefixes and joins data across Named Graphs to identify dependencies mentioned in vulnerability descriptions.

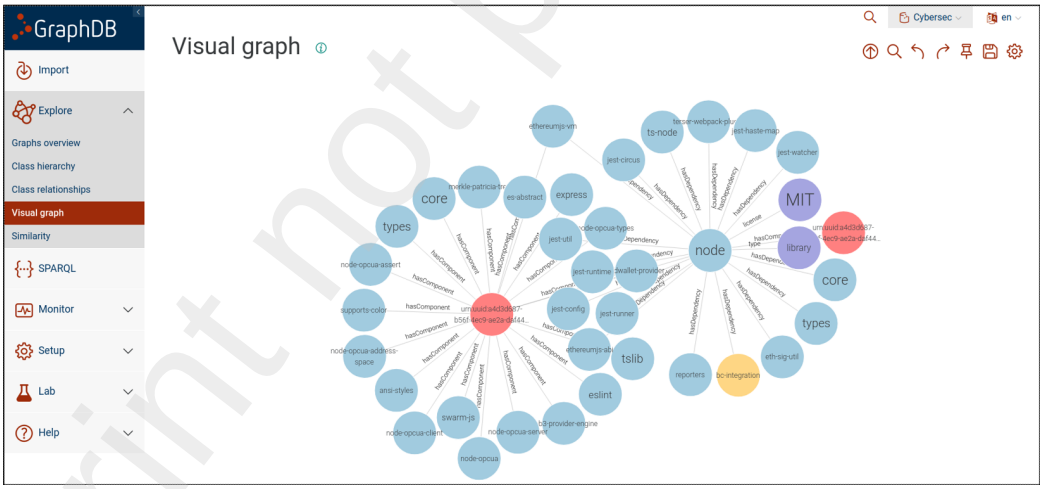


Figure 16: This figure presents a visual Knowledge Graph in GraphDB, illustrating relationships between software components, libraries, dependencies, and licenses. Nodes represent entities such as libraries and components, while edges denote relationships like hasComponent and hasDependency, providing an intuitive overview of the software supply chain structure.

Listing 1: SPARQL query for answering the competency question *Which import ID I can reach for a vulnerability defined in a NVD database? Vulnerabilty: CVE-2025-23211. We do not know in which place of the hierarchy this package lives, so we have to go upwards and downwards in the hierarchy*

```
PREFIX software_vocab: <http://ontologies.siemens-energy.com/software/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX se_sec_nvd_vocab: <https://ontology.siemens-energy.com/nvd/>
PREFIX cve: <https://data.siemens-energy.com/cybersec/cve/>

SELECT ?vulnerable_package ?downward_dependecy ?upward_dependency
WHERE {
    ?vulnerable_package rdf:type software_vocab:Software_package .
    ?vulnerable_package se_sec_nvd_vocab:hasVulnerability cve:CVE-2025-23211 .
    ?vulnerable_package software_vocab:hasDependency+ ?downward_dependecy .
    ?upward_dependency software_vocab:hasDependency+ ?vulnerable_package .
}
```

vulnerable_package	downstream_dependency	upward_dependency
Jinja2	Babel	Flask
Jinja2	MarkupSafe	Flask

Table 1: Response to query from listing 1.

Listing 2: SPARQL query for answering the competency question *Which toplevel imports I have in my Python environment (Python.json)?*

```
PREFIX software_vocab: <http://ontologies.siemens-energy.com/software/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX se_sec_nvd_vocab: <https://ontology.siemens-energy.com/nvd/>
PREFIX cve: <https://data.siemens-energy.com/cybersec/cve/>

SELECT ?package ?downward_dependecy ?upward_dependency
WHERE {
    ?package rdf:type software_vocab:Software_package .
    ?package software_vocab:hasDependency ?downward_dependecy .
    FILTER NOT EXISTS {
        ?upward_dependency software_vocab:hasDependency ?package .
  }
}
```

4. Discussion

This work presents a prototype pipeline for the generation of an EKG that represents cybersecurity information from a diverse set of data sources.

vulnerable_package	downward_dependency
Jinja2	Babel
Jinja2	MarkupSafe

Table 2: Response to query from listing 2. Only a fragment of the obtained results is shown.

In order to integrate the data, a common data model has been developed, the Cybersecurity Ontology Network, which provides a reusable language for annotating entities from the cybersecurity domain. A common data model facilitates the automation of the mapping process: such automations have been implemented through a set of reusable, declarative mappings, making a reproducible pipeline for the generation of RDF data that can be integrated in the company's internal CI/CD processes.

After a thorough review of the data sources, it was concluded that new ontologies needed to be developed. This process was facilitated by the fact that the data sources were structured and even included an already existing specification in the case of SBOM files (CycloneDX). CycloneDX represents an industry standard for package information that acts as a "backbone" to integrate the rest of cybersecurity information. Therefore, by providing a translation from the standard to a re-usable and live ontology, we offer the community a valuable tool for data integration.

The use of the W3C open standards RDF, OWL and SPARQL ensures the technical interoperability and avoid product lock-in, additionally setting the stage for the semantic interoperability provided by the newly created ontologies. This is a future-proof architectural choice, since it will allow for new data, unpredicted data sources to be processed by the pipeline, like module description files from other languages (e.g. Maven files in Java programming) or the vulnerability files from other repositories (e.g. European Union Vulnerability Database [46]).

The use of Semantic Web technologies also allows for the discovery of links between the integrated nodes. In this case, such a discovery process has been performed by parsing the content of the vulnerability description looking for package names extracted from SBOM files. This syntactic discovery, albeit reasonably performant, requires manual validation to reject links to packages with commons names (e.g. "click"). It is expected that in future developments a new entity reconciliation module will be implemented, based on more sophisticated link detection methods that exploit Machine Learn-

ing. Furthermore, best practices for creating URIs will be documented and included in the Knowledge Mapper (e.g. [47]), and a SHACL-based [48] data quality module will be added to production workflows.

The proposed EKG architecture is inherently scalable, both in terms of data volume and system complexity. The automated data ingestion pipeline transforms diverse structured sources—including JSON vulnerability reports (e.g., Amazon Inspector), software bill of materials (e.g., CycloneDX), and dependency manifests —into a unified semantic format without manual intervention, enabling continuous and high-throughput data integration. Furthermore, the modular design of the underlying ontology network allows for incremental expansion without impacting performance, as sub-ontologies can be developed, maintained, and reasoned over independently. The approach has been validated through deployment at Siemens Energy, where it successfully processes and integrates cybersecurity data at enterprise scale, demonstrating robustness and responsiveness in an operational setting. The extensibility of the proposed EKG framework is enabled by its modular ontology design, standards-based architecture, and flexible data integration pipeline. New data sources—such as emerging SBOM formats, vulnerability databases, or internal security tools—can be incorporated by defining mappings to the existing ontology without altering the system's core. The ontology network itself can be extended with additional classes, properties, or aligned with external vocabularies, allowing the system to evolve as cybersecurity standards and requirements change. This design ensures the framework remains adaptable to future threat models, compliance needs, and enterprise contexts beyond its initial deployment.

In the current prototype, even though the data are integrated, it is only available for consumption by SE users through technical interfaces. In the future, other interfaces are expected to be provided, like customized graph visualizations, BI dashboards, or LLM based Graph-RAG (Retrieval Augmented Generation) implementations such as GraphDB's Talk To Your Graph [49]. In the case of graph visualizations, the problem of building tailored interfaces for data consumers, which add value to the company's workflows, is a challenge that requires considerable investment.

Once the system is deployed to production, other aspects of it will become valuable. For example, in its current form, the prototype already complies with the content-related FAIR principles (F2, F3, I1, I2, I3, R1, R1.1, R1.2, R1.3). However, the infrastructure related principles (A1, A1.1, A1.2, A2) will be implemented in a company-wide infrastructure deployment. The de-

ployment to production will also allow other developers to contribute with more competency questions that will also require the extension of the model (e.g. "Which component prevents that this artifact can be updated to a version without known vulnerabilities with minimal changes?", "Will an upgrade to vulnerability free component of the project change the license conditions?".)

5. Conclusions

Current companies are moving from an Application-Centric Architecture to a Data-Centric Architecture, building information systems that seamlessly integrate information in future-proof, flexible and interoperable platforms. By doing so, companies comply with FAIR principles.

SE has developed a prototype for integrating information about cybersecurity from different sources: SBOM files, NVD files, and AWS ASFF files. This prototype includes a Knowledge Mapper that exploits semantic technologies (RDF, SPARQL, OWL) to integrate the data, based on the Cybersecurity Ontology Network to provide a common data model.

This methodology should be useful for companies involved in similar processes, and in case cybersecurity information is involved, the data model should provide a reusable base for data integration.

CRediT authorship contribution statement

Mikel Egaña Aranguren: Conceptualization, Methodology, Software, Validation, Investigation, Data Curation, Writing - Original Draft.

Jesualdo Tomás Fernández-Breis: Conceptualization, Methodology, Investigation, Supervision, Resources, Project administration, Funding acquisition, Writing - Original Draft.

Bidane Leon Balentzia: Software, Validation, Data Curation, Writing - Original Draft.

Markus Rompe: Conceptualization, Validation, Writing - Review & Editing.

Alexander García Castro: Conceptualization, Supervision, Resources, Project administration, Funding acquisition, Writing - Review & Editing.

Declaration of competing interest

The authors declare no conflict of interest.

Acknowledgments

This work has been funded by Siemens Energy.

Data availability

The data and the ontology network is available at a GitHub repository: `https://github.com/tecnomod-um/CybersecurityOntologyNetwork`.

URI Prefixes

Prefix	URI
dvu	`http://data.siemens-energy.com/cve/`
vul	`http://ontology.siemens-energy.com/cve/`
dsbom	`http://data.siemens-energy.com/sbom/`
sbom	`http://ontology.siemens-energy.com/sbom/`
dlib	`http://data.siemens-energy.com/softwarelibrary/`
lib	`http://ontology.siemens-energy.com/softwarelibrary/`

References

[1] R. Anderson, T. Moore, The economics of information security, Science 314 (5799) (2006) 610–613. arXiv:https://www.science.org/doi/pdf/10.1126/science.1130992, doi:10.1126/science.1130992.
URL `https://www.science.org/doi/abs/10.1126/science.1130992`

[2] L. A. Gordon, M. P. Loeb, The economics of information security investment, ACM Trans. Inf. Syst. Secur. 5 (4) (2002) 438–457. doi:10.1145/581271.581274.
URL `https://doi.org/10.1145/581271.581274`

[3] R. Böhme, G. Schwartz, Modeling cyber-insurance: Towards a unifying framework, in: Workshop on the Economics of Information Security, 2010.
URL `https://api.semanticscholar.org/CorpusID:14172008`

[4] C. Mavani, H. Mistry, R. Patel, A. Goswami, The role of cybersecurity in protecting intellectual property, International Journal on Recent and Innovation Trends in Computing and Communication 12 (2024) 529–538.

[5] ENISA Threat Landscape 2024 — ENISA (3 2025).
URL https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024

[6] Amazon AWS, Amazon Inspector (2025).
URL https://aws.amazon.com/inspector/

[7] CycloneDX, CycloneDX (2025).
URL https://cyclonedx.org/

[8] D. McComb, Software Wasteland: How the Application-centric Mindset is Hobbling Our Enterprises, Technics Publications, 2018.
URL https://books.google.es/books?id=_6JutAEACAAJ

[9] D. McComb, The Data-centric Revolution: Restoring Sanity to Enterprise Information Systems, Technics Publications, 2019.
URL https://books.google.es/books?id=5XuIxwEACAAJ

[10] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Computing Surveys (Csur) 54 (4) (2021) 1–37.

[11] J. M. Gomez-Perez, J. Z. Pan, G. Vetere, H. Wu, Enterprise knowledge graph: An introduction, in: Exploiting linked data and knowledge graphs in large organisations, Springer, 2017, pp. 1–14.

[12] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The FAIR guiding principles for scientific data management and stewardship, Scientific Data 3 (2016) 160018. doi:10.1038/sdata.2016.18.

[13] Z. Syed, A. Padia, M. L. Mathews, T. Finin, A. Joshi, et al., Uco: A unified cybersecurity ontology, in: Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security, 2016, pp. 195–202.

[14] B. A. Mozzaquatro, R. Jardim-Goncalves, C. Agostinho, Towards a reference ontology for security in the internet of things, in: 2015 IEEE International Workshop on Measurements & Networking (M&N), IEEE, 2015, pp. 1–6.

[15] N. Scarpato, N. D. Cilia, M. Romano, Reachability matrix ontology: a cybersecurity ontology, Applied Artificial Intelligence 33 (7) (2019) 643–655.

[16] J. A. Wang, M. Guo, Ovm: an ontology for vulnerability management, in: Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies, 2009, pp. 1–4.

[17] A. Shaked, N. Messe, T. Melham, Modelling tool extension for vulnerability management, in: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, 2024, pp. 56–60.

[18] B. B. Duarte, R. de Almeida Falbo, G. Guizzardi, R. Guizzardi, V. E. S. Souza, An ontological analysis of software system anomalies and their associated risks, Data & Knowledge Engineering 134 (2021) 101892.

[19] J. Yin, W. Hong, H. Wang, J. Cao, Y. Miao, Y. Zhang, A compact vulnerability knowledge graph for risk assessment, ACM Transactions on Knowledge Discovery from Data 18 (8) (2024) 1–17.

[20] Y. Qi, Z. Gu, A. Li, X. Zhang, M. Shafiq, Y. Mei, K. Lin, Cybersecurity knowledge graph enabled attack chain detection for cyber-physical systems, Computers and Electrical Engineering 108 (2023) 108660.

[21] E. Gilliard, J. Liu, A. A. Aliyu, Knowledge graph reasoning for cyber attack detection, IET Communications 18 (4) (2024) 297–308.

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

[23] I. Mouiche, S. Saad, Entity and relation extractions for threat intelligence knowledge graphs, Computers & Security 148 (2025) 104120.

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

[25] R. Ghosh, H.-M. von Stockhausen, M. Schmitt, G. M. Vasile, S. K. Karn, O. Farri, Cve-llm: Ontology-assisted automatic vulnerability evaluation using large language models, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 39, 2025, pp. 28757–28765.

[26] Z. Shi, H. Li, D. Zhao, C. Pan, Research on quality assessment methods for cybersecurity knowledge graphs, Computers & Security 142 (2024) 103848.

[27] X. Zhao, R. Jiang, Y. Han, A. Li, Z. Peng, A survey on cybersecurity knowledge graph construction, Computers & Security 136 (2024) 103524.

[28] Siemens Energy, Siemens Energy (2025).
URL https://www.siemens-energy.com/

[29] W3C, RDF 1.2 Concepts and Abstract Syntax (2025).
URL https://www.w3.org/TR/rdf12-concepts/

[30] W3C, OWL 2 Web Ontology Language Document Overview (Second Edition) (2025).
URL https://www.w3.org/TR/owl2-overview/

[31] W3C, SPARQL 1.2 Query Language (2025).
URL https://www.w3.org/TR/sparql12-query/

[32] NIST, CVE JSON schema (2025).
URL https://csrc.nist.gov/schema/nvd/feed/1.1-Beta/CVE_JSON_4.0_min_1.1_beta.schema

[33] NIST, JSON Schema for Common Vulnerability Scoring System version 2.0 (2025).
URL https://csrc.nist.gov/schema/nvd/feed/1.1-Beta/cvss-v2.0_beta.json

[34] NIST, JSON Schema for Common Vulnerability Scoring System version 3.x (BETA) (2025).

URL https://csrc.nist.gov/schema/nvd/feed/1.1-Beta/cvss-v3.x_beta.json

[35] Amazon AWS, Amazon AWS Security Hub (2025).
URL https://aws.amazon.com/security-hub/

[36] Amazon AWS, Amazon AWS Security Finding Format (2025).
URL https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-findings-format.html

[37] ECMA international, ECMA 424 (2025).
URL https://ecma-international.org/publications-and-standards/standards/ecma-424/

[38] MITRE Corporation, CVE (2025).
URL https://www.cve.org/

[39] MITRE Corporation, MITRE Corporation (2025).
URL https://www.mitre.org/

[40] NVD, NVD (2025).
URL https://nvd.nist.gov/

[41] NVD, NVD feeds (2025).
URL https://nvd.nist.gov/vuln/data-feeds

[42] RML, YARRRML: A human readable text-based representation for declarative Linked Data generation rules (2025).
URL https://rml.io/yarrrml/

[43] W3C, Data Catalog Vocabulary (DCAT) (2025).
URL https://www.w3.org/TR/vocab-dcat-3/

[44] W3C, PROV ontology (2025).
URL https://www.w3.org/TR/prov-o/

[45] W3C, SPARQL 1.2 Protocol (2025).
URL https://www.w3.org/TR/sparql12-protocol/

[46] European Union Agency for Cybersecurity, European Union Vulnerability Database (2025).
URL https://euvd.enisa.europa.eu/

[47] UK , UK Government (2025).
URL DesigningURIsetsfortheUKpublicsector

[48] W3C, Shapes Constraint Language (SHACL) (2025).
URL https://www.w3.org/TR/shacl/

[49] Ontotext, GraphDB Talk To Your Graph (2025).
URL https://graphdb.ontotext.com/documentation/11.0/talk-to-graph.html#