Deliverable D2.11

Certification Graph – v2

Editor(s):	Verena Geist, Stefan Schöberl
Responsible Partner:	Software Competence Center Hagenberg GmbH
Status-Version:	Final-v1.0
Date:	30.01.2026
Type:	OTHER
Distribution level (SEN, PU):	PU

DRAFT

Project Number:	101120688
Project Title:	EMERALD

Title of Deliverable:	Certification Graph – v2
Due Date of Delivery to the EC:	31.01.2026

Workpackage responsible for the Deliverable:	WP2 – Methodology for Knowledge Extraction
Editor(s):	Verena Geist, Stefan Schöberl (SCCH)
Contributor(s):	Angelika Schneider (FHG), Hussain Hussain (KNOW)
Reviewer(s):	Angela Fessl (KNOW) Cristina Martínez, Juncal Alonso (TECNALIA)
Approved by:	All Partners
Recommended/mandatory readers:	WP2, WP3

Abstract:	EMERALD aims to integrate evidence collected at different levels of the cloud service into a single graph-based structure, the *Certification Graph*. This document describes the final version of the graph, i.e. its schema (or ontology), with semantically linked and combined evidence. The development mainly involves work of T2.1 and T2.6, but also inputs of T2.2, T2.3, T2.4, T2.5, and T3.1 are considered.
Keyword List:	Knowledge graph schema, ontology extensions, security features, knowledge integration, combined evidence analysis, ontology refinement.

Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 4.0 International (**CC BY-SA 4.0 DEED** https://creativecommons.org/licenses/by-sa/4.0/)
Disclaimer	Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. The European Union cannot be held responsible for them.

DRAFT

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	03.12.2025	First draft version, key information, TOC, and changes	Verena Geist (SCCH)
v0.2	04.12.2025	Update of executive summary and introduction	Verena Geist (SCCH)
v0.3	10.12.2025	Update of technical description, including ontology extensions, delivery and usage, and future development	Stefan Schöberl (SCCH)
v0.4	15.12.2025	Update of functional description, TLS example, and conclusions	Verena Geist (SCCH)
v0.5	18.12.2025	Description of the AIOX tool added	Verena Geist (SCCH), Hussain Hussain (KNOW)
v0.6	08.01.2026	Revised version and minor updates	Angelika Schneider (KNOW), Verena Geist (SCCH)
v0.7	12.01.2026	Internal review	Angela Fessl (KNOW)
v0.8	14.01.2026	Revision of the document based on the internal review	Verena Geist (SCCH) Stefan Schöberl (SCCH)
v0.9	23.01.2026	Final reviewed version	Cristina Martínez, Juncal Alonso (TECNALIA)
v1.0	30.01.2026	Submitted to the European Commission	Cristina Martínez, Juncal Alonso (TECNALIA)

DRAFT

(cc) BY-SA

Table of contents

Terms and abbreviations 6
Executive Summary 7
1 Introduction 8
1.1 About this Deliverable 8
1.2 Document structure 9
1.3 Updates from D2.10 9
1.4 Technological advances from the MEDINA project 10
2 The CertGraph ontology 12
2.1 Functional description 12
2.2 Technical description 15
2.2.1 Architecture 15
2.2.2 Subcomponents description 17
2.2.3 Technical specifications 22
2.3 Delivery and usage 23
2.3.1 Download 23
2.3.2 Package information 23
2.3.3 Instructions for use 23
2.3.4 Licensing information 24
2.4 Future development of the ontology 24
3 Refined illustrative example of modelling and combining evidence information for the used TLS Version 25
4 AIOX – AI-assisted Ontology Extension Tool for CertGraph Refinement 28
4.1 Functional description 28
4.2 Technical description 29
4.2.1 Architecture 29
4.2.2 Technical specifications 29
4.3 Delivery and usage 30
4.3.1 Download 30
4.3.2 Instructions for use 30
4.3.3 Licensing information 31
4.4 Future developments 32
5 Conclusions 33
6 References 34

List of tables

TABLE 1. OVERVIEW OF DELIVERABLE UPDATES WITH RESPECT TO D2.10 [2] 9
TABLE 2. REQ.01 - FORMAL LANGUAGE 12

DRAFT

(cc) BY-SA

TABLE 3. REQ.02 - CLEAR CONCEPTUALIZATION 12
TABLE 4. REQ.03 - HIERARCHICAL STRUCTURE OF CONCEPTS 12
TABLE 5. REQ.04 - REASONING AND CONSISTENCY CHECKING 13
TABLE 6. REQ.05 - INTEROPERABILITY AND EXTENSIBILITY 13
TABLE 7. REQ.06 - DOCUMENTATION AND ANNOTATION 13
TABLE 8. REQ.07 - VERSIONING 13
TABLE 9. OVERVIEW AND DESCRIPTION OF THE PACKAGE STRUCTURE FOR THE CERTGRAPH ONTOLOGY 23

List of figures

FIGURE 1. EMERALD COMPONENT OVERVIEW DIAGRAM [13] 15
FIGURE 2. UPDATED DESIGN OF THE CERTGRAPH ONTOLOGY 16
FIGURE 3. EXCERPT OF THE EVIDENCE SUB-ONTOLOGY 17
FIGURE 4. EXCERPT OF THE FRAMEWORK SUB-ONTOLOGY 18
FIGURE 5. EXCERPT OF THE FUNCTIONALITY SUB-ONTOLOGY 18
FIGURE 6. EXCERPT OF THE PROPERTIES SUB-ONTOLOGY 19
FIGURE 7. EXCERPT OF THE SECURITY SUB-ONTOLOGY 19
FIGURE 8. EXCERPT OF THE CLOUD ONTOLOGY EXTENSION 20
FIGURE 9. EXCERPT OF THE APPLICATION ONTOLOGY EXTENSION 21
FIGURE 10. EXCERPT OF THE ML ONTOLOGY EXTENSION 21
FIGURE 11. EXCERPT OF THE DOCUMENT ONTOLOGY EXTENSION 22
FIGURE 12. EXCERPT OF THE HUMAN RESOURCES ONTOLOGY EXTENSION 22
FIGURE 13. CLASSES (RECTANGLES) AND INSTANCES (HEXAGONS) FOR THE TLS EXAMPLE, SHOWING EVIDENCE FOUND IN SOURCE CODE (IMPLEMENTED) AND CORRESPONDING EVIDENCE IN A DOCUMENT (SPECIFIED) REGARDING TRANSPORT ENCRYPTION, WHICH CAN BE USED TO VERIFY CRY-02 FROM BSI C5:2020 (ADAPTED FROM [9]) 25
FIGURE 14. MODULAR PIPELINE OF THE AIOX TOOL 29
FIGURE 15. EXAMPLE METRIC: TRANSPORTENCRYPTIONENABLED.YAML 30
FIGURE 16. INTERACTIVE PROMPT: NEW TERM IDENTIFICATION 31
FIGURE 17. INTERACTIVE PROMPT: PROPERTY SUGGESTION 31

DRAFT

(cc) BY-SA

Terms and abbreviations

AI	Artificial Intelligence
AIOX	AI-assisted Ontology Extension
AI-SEC	AI Security Evidence Collector
AMOE	Assessment and Management of Organizational Evidence
BSI	Bundesamt für Sicherheit in der Informationstechnik
BSI C5	BSI Cloud Computing Compliance Criteria Catalogue
CertGraph	Certification Graph
CICD	Continuous Integration Continuous Delivery
Codyze	Static Code Analyzer
EC	European Commission
eknows	Platform for software analysis
eknows-e3	Extractor component developed in the context of EMERALD
GA	Grant Agreement to the project
HR	Human Resources
HTTP	Hypertext Transfer Protocol
ID	Identity
KR	Key Result
MEDINA	GA 952633. Predecessor project of EMERALD
ML	Machine Learning
MLM	Masked Language Models
NLP	Natural Language Processing
OTP	One-time password
OWL	Web Ontology Language
PDF	Portable Document Format
Protobuf	Protocol Buffers
RDF	Resource Description Framework
RCM	Repository of Controls and Metrics
SSO	Single Sign-On
SPARQL	SPARQL Protocol and RDF Query Language
SWRL	Semantic Web Rule Language
TLS	Transport Layer Security
UI	User Interface
URI	Uniform Resource Identifier
WP	Work Package
XML	Extensible Markup Language
YAML	Yet Another Markup Language

DRAFT

(cc) BY-SA

Executive Summary

This deliverable describes the final version of the central *Certification Graph* schema (i.e., the *CertGraph* ontology) for storing evidence in a graph-based format and the refinement of the initial work on designing the *CertGraph* ontology in D2.1 [1] and describing its interim version in D2.10 [2]. This ontology serves as a common structure for semantically linked and combined evidence that is filled by all evidence extraction components of WP2.

By developing the *CertGraph* ontology, this deliverable contributes to the key result CERTGRAPH (KR2) of the EMERALD project to provide a unified graph-based model of the target of evaluation at different layers of the cloud service. Following a knowledge graph-based approach, in EMERALD, the ontology for storing and linking heterogenous evidence information is developed in WP2, and the model is further implemented in the *Evidence Store* described in D3.4 [3] in WP3.

The main result of this deliverable is a uniform graph-based model that

(i) consolidates all extracted evidence information,
(ii) enables the retrieval of combined evidence by aggregating individual pieces of information to a higher-level viewpoint,
(iii) maintains traceability back to different information sources and extraction tools,
(iv) provides all required concepts for resource types and security features to assess certification-relevant security metrics, and
(v) ensures scalable and consistent knowledge evolution.

This document starts by giving a short introduction and highlighting updates from D2.10 [2], the previous version of this deliverable, as well as technological advances from the predecessor project MEDINA. The main part provides the functional and technical descriptions of the ontology, including its sub-ontologies and extensions to support the holistic approach of evidence collection. These descriptions are complemented by information on delivery and usage, as well as on limitations and future work. Then, a refined example of modelling and combining evidence information for Transport Layer Security (TLS) encryption from different sources illustrates the purpose and innovation of the ontology. Additionally, a newly developed AI-assisted tool is presented, including its functional and technical descriptions and the instructions for use. This tool guides experts in consistently extending the *CertGraph* ontology when developing new security metrics. The document finally concludes with a short summary.

DRAFT

(cc) BY-SA

1 Introduction

The *CertGraph* ontology, previously described in D2.1 [1] and D2.10 [2], is a central graph-based model to support certification by bridging different layers and sources of extracted information, called evidence, from a cloud service. For this purpose, the ontology provides a highly structured, formal representation of a set of concepts (or classes) and their relationships and properties within the cloud service certification domain. It is created using a formal language, i.e. the Web Ontology Language (OWL), which supports complex expressions and logical inferences, including constraints, class hierarchies, and more. The main purpose of an ontology is to support knowledge sharing and reuse through structured domain knowledge, as well as reasoning about the entities within the domain. In EMERALD, the *CertGraph* ontology enables harmonization of evidence gathering and assessment. Security controls defined in different schemas or catalogues are assigned to ontological concepts and those ontological types will be further used in metric definitions.

For automated compliance tools to work, suitable evidence needs to be extracted and linked. The evidence extractors developed in the EMERALD project, described in D2.3 [4], D2.5 [5], D2.7 [6], and D2.9 [7], extract and provide suitable evidence from:

(i) the source code of services, often written in different programming languages, such as Java, Go, or Python (*Codyze* and *eknows-e3*),
(ii) relevant parts of legal and policy documents, such as requirement or architecture documents (*AMOE*),
(iii) applied machine learning (ML) models with respect to various criteria, such as robustness, fairness, and explainability (*AI-SEC*), and
(iv) the virtual infrastructure, such as virtual machines, containers, or storage as well as runtime information, such as configurations, log files, and provenance information (*Clouditor-Discovery* and *Codyze-Provenance*).

The *CertGraph* ontology with its respective extensions described in this document is a central tool to bridge those different layers and sources of evidence. Furthermore, security metrics are linked to structured, reusable ontology knowledge to ensure scalable and consistent evolution. Therefore, the ontology defines a vocabulary for mapping between the properties that shall be measured and the respective gathering of adequate evidence. It allows to aggregate individual aspects and fragments of information to a higher-level viewpoint of combined evidence, not previously detectable by a single tool.

1.1 About this Deliverable

This document aims to describe the *CertGraph* ontology for modelling evidence information in the cloud service certification domain as a common structure for semantically linked and combined evidence. It consists of a core ontology and several sub-ontologies for capturing security features as well as domain *concepts* and *relationships* of different extensions. In this deliverable, the structure and the main concepts of the extensions to consolidate all extracted evidence information in terms of taxonomies is presented. In addition, *properties* are discussed which maintain traceability back to different information sources and extraction tools. For better illustration, we base the explanations on an example which uses one selected security criteria "encryption of data for transmission", which is specified in the BSI C5:2020[1] (CRY-02).

The *CertGraph* ontology, designed in Task 2.1, represents the basis for integrating and instantiating the graph-based model in the *Evidence Store* component [3] in Task 3.1. It is also

[1] https://www.bsi.bund.de/dok/13368652

(cc) BY-SA

the foundation for analysing the semantic information and context of the heterogeneous evidence information in Task 2.6 to enable the retrieval of combined evidence.

Another important aspect of the document is to semantically describe how specific *Resource Types* are related to *Security Features*, which are essential concepts to assess certification-relevant security metrics in Task 3.4. The newly developed *AI-assisted Ontology Extension* (AIOX) tool introduces semi-automated support for identifying, validating, and integrating new ontology concepts derived from metric definitions. Its main motivation is to reduce manual modelling effort while preserving expert control over ontology evolution, ensuring scalability and semantic quality.

1.2 Document structure

The document is structured as follows.

Section 2 provides the functional and technical descriptions of the *CertGraph* ontology, as well as information on its delivery and usage. Details on the sub-ontologies and extensions for the different cloud service layers are presented, e.g., for extracted evidence from source code, from policy documents, from ML models, and from cloud runtime environments. We further discuss refinements of concepts for combining evidence and supporting traceability, as well as for security features to assess (new) security metrics.

In Section 3, the illustrative example originally outlined in D2.1 [1] for modelling and combining extracted evidence information from different sources is refined based on new developments.

Section 4 describes an AI-driven ontology extension tool that supports expert-controlled refinement of the *CertGraph* ontology from security metric definitions.

Section 5 ends up with the conclusions of this deliverable.

1.3 Updates from D2.10

This deliverable evolves from D2.10 [2], and with the ultimate goal of making the document self-contained and easier to follow, part of the content comes from D2.10 [2] since it has not changed, and other parts are new. To simplify tracking progress and updates from the previous version, Table *1* shows a summary of the main changes and additions to each section of the document.

Table 1. Overview of deliverable updates with respect to D2.10 [2]

Section	Changes
Overall document	• Deleted previous Section 2 *Recap of the initial draft of the CertGraph ontology and changes*, because all changes are now reflected in Sub-section 1.3 • Updated TOC and other tables • Updated terms and abbreviations • Updated executive summary • Updated references
1 Introduction	• Updated outline of this deliverable • Added new Sub-section 1.3 *Updates from D2.10* • Added new Sub-section 1.4 *Technological advances from the MEDINA project*

(cc) BY-SA

Section	Changes
2 *The CertGraph ontology*	• Updated heading and description (from interim to final version of the ontology) • Updated functional description to reflect the current development status of the *CertGraph* ontology • Updated requirement status • Updated architecture diagram to also reflect new extensions and changes in the ontology structure • *CertificationTarget* was renamed to *TargetOfEvaluation* in the *Evidence* sub-ontology • Added new organizational taxonomy (*Human Resources*) as additional extension • Refinement of the *Cloud*, *ML*, and *Document* ontologies based on security metrics • Updated and extended technical specifications • Updated delivery and usage information • Updated and extended package information • Updated future development work
3 Refined illustrative example of modelling and combining evidence information for the used TLS version	• *CertificationTarget* was renamed to *TargetOfEvaluation* in the illustrative example • Added description of the control involved in the example
4 AIOX – AI-assisted Ontology Extension Tool for *CertGraph* Refinement	• Added section to describe the newly developed AI-assisted tool that guides experts in (semi-)automatically extending the *CertGraph* ontology
5 Conclusions	• Updated conclusions on the *CertGraph* ontology

1.4 Technological advances from the MEDINA project

EMERALD builds upon the outcomes of the MEDINA project[2]: *Security framework to achieve a continuous audit-based certification in compliance with the EU-wide cloud security certification scheme* (GA 952633). While MEDINA only reached TRL5, the goal of EMERALD is to enhance the system and elevate its TRL from prototype (TRL5) to product (TRL7). This will be achieved by:

- making AI and high-level technologies approachable for non-experts,
- increasing the abstraction level in evidence extraction across various layers of the cloud service (infrastructure, code, business processes) in an easy-to-use and understandable way, and
- providing trustworthy support for continuous auditing processes.

To support these objectives, the different sub-ontologies and extensions of the *CertGraph* ontology together form a unified source of types in the cloud service certification domain. The *Core* ontology, together with the *Security Feature* ontology, builds the foundation of the ontology and contains base classes and properties. Specifically, *Security Feature* models different security related concepts. The extensions are built on top of this foundation, and each extension models the evidence gathered by a different type of extractor (i.e., *eknows-e3* /

[2] https://medina-project.eu/

(cc) BY-SA

Codyze, *AMOE*, *AI-SEC*, and *Clouditor-Discovery*). This approach has the major advantage that security metrics can be defined for abstract resource types and/or security features, while the extractor tools can agnostically gather evidence for these abstract concepts as well.

The *CertGraph* ontology is based on the *Cloud Property Graph* [8] ontology from MEDINA, which proposes a vendor-independent ontology of cloud resources and related security features. Key concepts from the *Cloud Property Graph* are reused to build *Core* (including reuse of *security features* and *functionality taxonomies*) and *Cloud* (extension and refinement of *cloud resources* and their properties). Major changes in EMERALD concern:

- modelling evidence and resources in *Core*, including further refinement,
- refining *Core* into smaller sub-ontologies to improve the ontology structure,
- adding security feature concepts and properties based on new metrics and pilot requirements,
- adding the specific extensions, in particular
    - *Application* to model and categorize applications including source code
    - *Document* to model and categorize textual information from (policy) documents and add corresponding properties
    - *ML* to model and categorize machine learning-relevant aspects
- metrics are not part of the ontology (as was the case in MEDINA).

Finally, the AIOX tool represents a technological advance within EMERALD by introducing AI-assisted, human-in-the-loop ontology engineering into the certification workflow. It goes beyond static ontology modelling by combining natural language processing, semantic embeddings, and masked language models to support the systematic identification, placement, and formalisation of new ontology concepts derived from security metrics. In contrast to manual or rule-based approaches, the tool dynamically analyses both metric semantics and existing ontology structure, providing context-aware recommendations while preserving expert control. This enables a scalable, consistent, and traceable evolution of the *CertGraph* ontology, addressing the key challenges of practical maintenance and time-intensive extension of the knowledge base.

DRAFT

(cc) BY-SA

2 The CertGraph ontology

In this section, we describe the final version of the *CertGraph* ontology, the schema of the envisaged *Certification Graph* designed to streamline security certification which integrates evidence from multiple sources [9].

2.1 Functional description

Motivation and scope. The foundation of the *Certification Graph* is an ontology to store and link evidence from different sources. We consider the complete stack from software to policies and enable the fusion of extracted evidence from different views. Its extensible structure is designed to accommodate multiple domains, including cloud security, ML models, documents, and source code. By providing an automated and systematic approach to extract evidence from different sources and build an ontology, the resulting *Certification Graph* aims to facilitate more effective security certification and compliance verification [9].

Requirements. In D2.1 [1], a list of requirements for developing the ontology was introduced. In the following, their respective implementation state (partially / fully /not implemented) is updated from D2.10 [2] and brief descriptions are given in Table 2 to Table 8.

Table 2. REQ.01 - Formal language

Field	Description
Requirement ID	REQ.01
Short title	Formal language
Description	The ontology should be defined using a formal language that allows for the expression of concepts, relationships, instances, and axioms.
Progress	Fully Implemented – 100 %

The ontology is defined using the Web Ontology Language (OWL)[3].

Table 3. REQ.02 - Clear conceptualization

Field	Description
Requirement ID	REQ.02
Short title	Clear conceptualization
Description	The ontology should provide a clear and comprehensive conceptualization of the domain it represents.
Progress	Fully Implemented – 100 %

The ontology extensions have been further refined, for application, ML, and document.

Table 4. REQ.03 - Hierarchical structure of concepts

Field	Description
Requirement ID	REQ.03
Short title	Hierarchical structure of concepts
Description	The ontology should support the creation of a hierarchical structure of concepts, allowing for subclass relationships and the organization of concepts into a taxonomy.
Progress	Fully Implemented – 100 %

[3] https://www.w3.org/OWL/

(cc) BY-SA

The sub-ontologies and extensions are modelled as taxonomies.

Table 5. REQ.04 - Reasoning and consistency checking

Field	Description
Requirement ID	REQ.04
Short title	Reasoning and consistency checking
Description	The ontology should be compatible with inference engines and allow for the definition of logical rules that enable automated reasoning about the concepts and their relationships.
Progress	Fully Implemented – 100 %

The selected modelling tool supports a reasoning component to derive new information based on rules and to detect inconsistencies in the ontology.

Table 6. REQ.05 - Interoperability and extensibility

Field	Description
Requirement ID	REQ1.05
Short title	Interoperability and extensibility
Description	The ontology should be developed in a way that ensures interoperability with other ontologies, facilitating data exchange and integration across different layers of a cloud service.
Progress	Fully Implemented – 100 %

The selected modelling tool supports the splitting of the ontology into multiple files for better structuring and linking of concepts using different namespaces, i.e., through different sub-ontologies and extension.

Table 7. REQ.06 - Documentation and annotation

Field	Description
Requirement ID	REQ.06
Short title	Documentation and annotation
Description	Comprehensive documentation and annotation of the ontology should be available.
Progress	Fully implemented – 100%

A README file in the *Collaborative Metrics Repository*[4] (see D3.4 [3]) gives an overview of the ontology and explains the necessary steps for contribution. The ontology files use annotations to include human-readable explanations of ontology concepts. The development progress of the ontology is tracked by using GitHub issues.

Table 8. REQ.07 - Versioning

Field	Description
Requirement ID	REQ.07
Short title	Versioning
Description	There should be a clear strategy for handling the releases of the ontology (e.g., annually, quarterly, or on demand) and how changes and new versions are announced.

[4] https://github.com/Cybersecurity-Certification-Hub/security-metrics/tree/main/ontology

(cc) BY-SA

Progress	Fully implemented – 100%

The used version control tool supports versioning of OWL files and collaboration. Main releases are contained in respective folders in the Repository[5]. Releases within the EMERALD project are marked with Git releases and for breaking changes a new version folder will be added. These releases are synchronised with the release of D2.10 (*Certification Graph* v1) [2] and with this deliverable (*Certification Graph* v2), and there will be a final one at the end of the project.

Innovation. Previous approaches [8] [10] [11] [12] that build on the notion of gathering evidence – from sources such as the cloud infrastructure – to demonstrate compliance to certain standards or regulations have two shortcomings. They perform a mapping to a structure described in an ontology to harmonize evidence gathered from various cloud providers and technologies, but they are not very comprehensive in terms of semantic modelling. For example, they mostly focus on cloud infrastructure resources, however, in a real-world certification scenario, many more resource types, such as source code, policy documents, or other data assets need to be assessed. Second, all previous approaches create different, independent kinds of evidence for each resource and store them into information silos, even if they describe the same aspect (e.g., configuration of encryption), but from different viewpoints.

The knowledge graph-based approach in EMERALD goes beyond these shortcomings: First, the *Certification Graph* aims to be a systematic approach to building an ontology for security certifications spanning the complete stack from infrastructure layer, source code, data to policies and procedures [9]. By providing a schema for storing and linking the heterogeneous evidence information, EMERALD can provide a unified view of the cloud service under certification. Second, by linking generic models at different levels of abstraction the *Certification Graph* enables the development and assessment of complex mapping rules. It provides a novel approach for the fusion of evidence coming from different views/sources of the same resource [9]. This approach allows to aggregate individual aspects and fragments of information to a higher-level of combined evidence, while providing support for traceability to information sources and extraction processes.

This way, the *Certification Graph* (i) serves as a common structure that is filled by all evidence extraction tools of WP2, (ii) can be leveraged by the assessment tools to measure security metrics relevant for certification, and (iii) can be systematically and consistently extended by combining AI-based semantic reasoning with a human-in-the-loop workflow, which makes the EMERALD solution approach outstanding and innovative. Assessing (partial) evidence from different sources also allows a qualitative statement about the accuracy of measured results for auditors and, furthermore, enables the comparison between specification (e.g., in policy documents) and implementation (e.g., in source code) of security features.

Fitting into overall EMERALD architecture. How the *CertGraph* ontology fits into the overall EMERALD architecture and how it is related with the other components was already described in D2.1 [1]. Figure 1 shows the EMERALD high-level architecture as a component diagram in its current state. There have been no major changes to the component diagram since D2.10 [2]: The extraction components of WP2 for collecting evidence, i.e., *AMOE*, *Codyze*, *eknows-e3*, *AI-SEC*, and *Clouditor-Discovery* are represented at the bottom part of Figure 1. They map the extracted information to the EMERALD evidence format using the terms described in the *CertGraph* ontology. The *CertGraph* ontology is a formal model (but not considered as a component itself) and defines the schema of the common graph-based structure using abstract types of the cloud certification domain. The raw evidence is then delivered to the *Evidence Store*

[5] https://github.com/Cybersecurity-Certification-Hub/security-metrics/tree/main/ontology/v1

(cc) BY-SA

[3], which implements the model as a component in WP3, and is used to assess the security metrics defined in the *Repository of Controls and Metrics* (*RCM*).

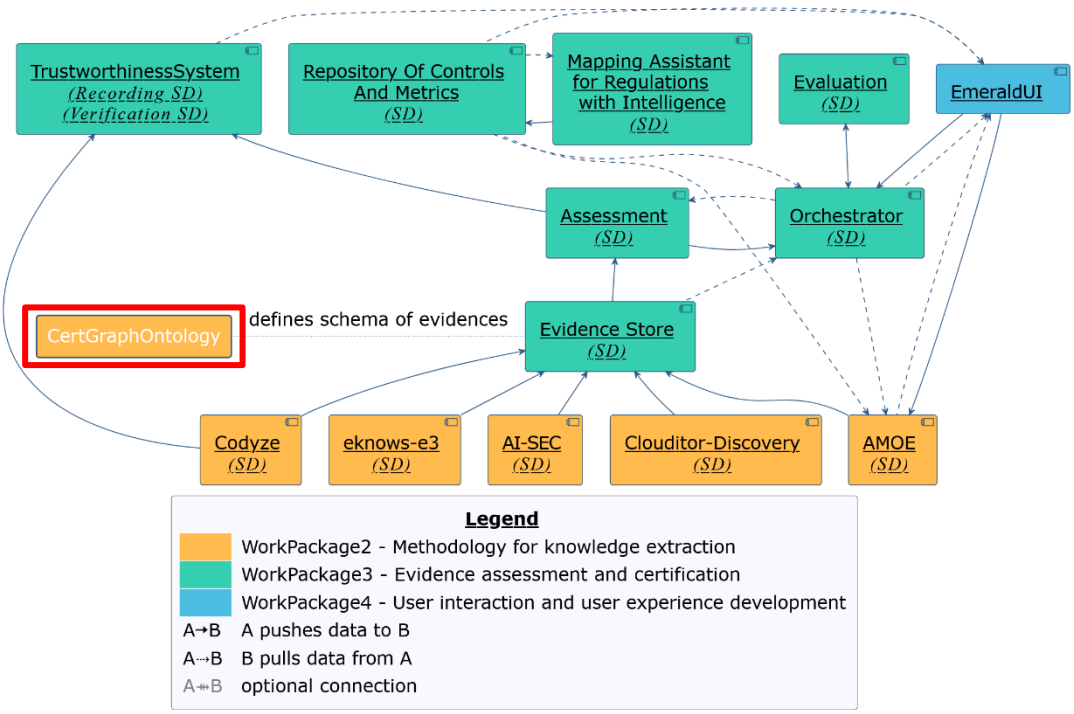

Figure 1. EMERALD component overview diagram [13]

Usage. In EMERALD, the *CertGraph* ontology is used for:

- Defining the schema of evidence to be stored in the *Evidence Store* (see D3.4 [3]).
- Preparing suitable evidence by the extraction components according to the ontology terms. The *tool Owl2proto*[6] is used to convert the modelled ontology to an appropriate *protobuf* schema, which can be directly used in different programming languages (see D2.1 [1]).
- Refining the modelled domains (in particular, for resource types and security features) based on new security metrics. The newly developed AIOX tool supports the systematic identification, placement, and formalisation of new ontology concepts derived from metric descriptions.

2.2 Technical description

The following subsections describe the technical details of the *CertGraph* ontology.

2.2.1 Architecture

The *CertGraph* ontology consists of multiple smaller ontologies. They include essential taxonomies (e.g., for resources, security features, or specific extensions) and define relationships between them (e.g., to describe which resource generally offers which security feature). The nodes along the whole hierarchy within the ontology can have (and inherit) relationships to other taxonomies.

[6] https://github.com/oxisto/owl2proto

(cc) BY-SA

As shown in Figure 2, five sub-ontologies form the *Core*: *Evidence*, *Framework*, *Functionality*, *Properties,* and *Security*. Extensions are built on top of the *Core* and each of them covers its own domain. We propose the following five extensions:

- A source code taxonomy (*Application*) to categorize and organize code elements based on their characteristics and functionalities.
- An organizational taxonomy (*Document*) to categorize and organize textual information from policy documents.
- An AI taxonomy (*ML*) to categorize and organize information extracted from ML models based on selected criteria.
- A cloud resource taxonomy (*Infrastructure*) to categorize and organize information extracted from cloud resources with application-specific runtime information.
- A taxonomy for human resources (*HR*) to categorize and organize information from (security) trainings, which constitutes a new extension that emerged from pilot requirements.

In addition, concepts defined in the *Machine Learning (ML)* and *Document* taxonomies share a common ancestor: *Data*. ML concepts are considered as *Data* because ML models represent a specific form of data. They encapsulate information derived from the processing of training data. An ML model contains structured data such as parameters, weights, or decision rules that can be utilized in machine learning-based processes. Thus, it is not an independent entity but a derived data form, like documents or tables. Categorizing ML as *Data* ensures semantic consistency and simplifies the ontology structure.

On the other hand, *Application* is treated as a separate concept from *Data* because it is functional and independent, although it is stored in source code files as text. Unlike ML model data, source code provides instructions that control and execute processes. Source code is not merely a representation of information but actively executable and essential for implementing and utilizing systems like ML models or other computational frameworks. This functional distinction justifies the classification of *Application* as a standalone concept outside of *Data*.

This modular approach also allows for further extension of the ontology by developing new extensions for other domains, if needed. For all sub-ontologies and new extensions, customized URIs are used to avoid interoperability issues when working with multiple namespaces and combing evidence information.

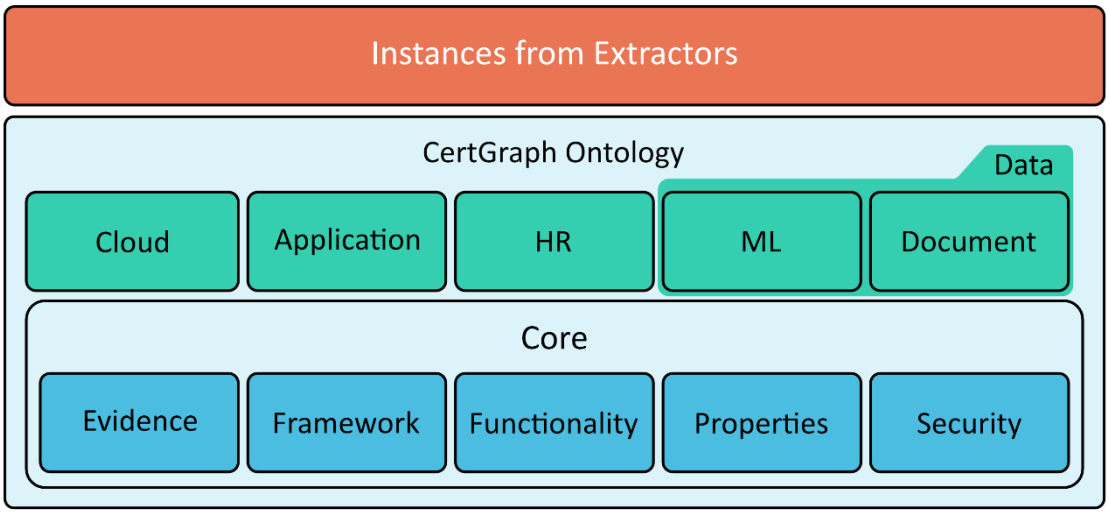

Figure 2. Updated design of the CertGraph ontology

(cc) BY-SA

2.2.2 Subcomponents description

This section presents the current state of the (sub-)ontologies. As already mentioned, some ontologies and extensions reuse and extend key concepts from the *Cloud Property Graph* [8], which was developed within the MEDINA project. The *Framework* and *Functionality* ontologies build upon the taxonomies of the same name defined in the *Cloud Property Graph*. Security-related aspects are derived from the *SecurityFeature* taxonomy, and the *Cloud* extension is based on the *CloudResource* taxonomy. They are modelled using Protégé[7] and stored as OWL/XML files.

Please note that some of the following screenshots show irregular word breaks; these are caused by the limitations of the tool used and cannot be adjusted or corrected within the software.

2.2.2.1 *Core* – a base ontology

Core forms the base of the *CertGraph* ontology and is composed of five ontologies. From a technical perspective, this ontology is just a wrapper around the five sub-ontologies by importing them. There are two use cases for this ontology: First, when developing extensions, this ontology must be imported into the extension to reuse concepts from *Core*. Second, when instantiating the ontology, *Core* describes the (base) structure of the resulting graph-based model.

2.2.2.1.1 Evidence – linking resources with security features

This sub-ontology models detected or extracted evidence regardless of the actual source. Each ***Evidence*** is connected to a *SecurityFeature*, to a ***Tool*** (to link the extraction tool for traceability), to a ***Resource*** (to store the detection point for traceability), and to a ***TargetOfEvaluation*** (to link to the related cloud service, for example). Furthermore, *Resource* has a connection to ***ResourceType*** (modelled as an enumeration type), to distinguish between specified and implemented behaviour.

Figure 3 shows an excerpt of the *Evidence* sub-ontology, i.e. the class hierarchy. Therefore, *SecurityFeature* is not shown in this figure (details can be found in Section 2.2.2.1.5). Further, connections (object and data properties) between classes are not shown in this diagram. An example of connecting various instances of those classes can be found in Section 3.

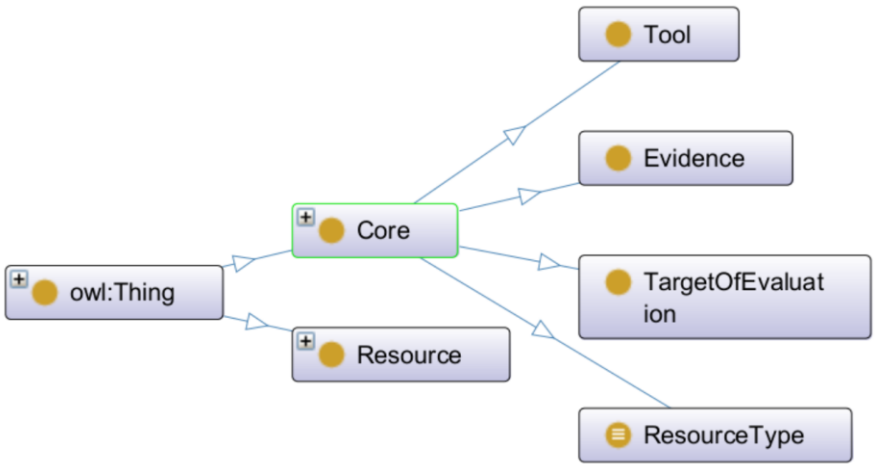

Figure 3. Excerpt of the Evidence sub-ontology

[7] https://protege.stanford.edu/

(cc) BY-SA

2.2.2.1.2 Framework – containing common types of software components

Common (high-level) types of software components are modelled in the *Framework* ontology (see Figure 4), which can be reused across different resources. This includes, for example, a ***HttpServer*** or a ***Logging*** component.

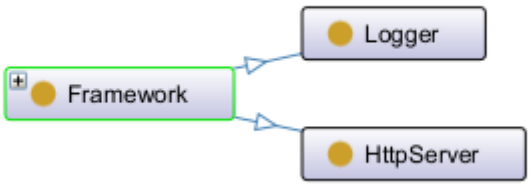

Figure 4. Excerpt of the Framework sub-ontology

2.2.2.1.3 Functionality – containing common data types

In addition to the high-level types (defined in *Framework*), smaller parts of software must be modelled. Also, in many parts of the *CertGraph* ontology, simple record types are needed. The *Functionality* ontology (see Figure 5) models all needed types, without restriction to a specific domain. For example, ***HttpEndpoint*** or ***HttpRequest*** are two classes in this ontology, which model smaller parts of software. In contrast, ***CipherSuite***, for example, is used as a record type and wraps the respective properties.

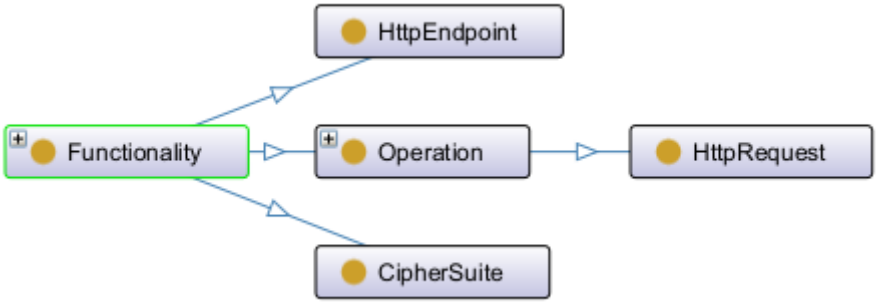

Figure 5. Excerpt of the Functionality sub-ontology

2.2.2.1.4 Properties – containing common object and data properties

Within the whole *CertGraph* ontology, classes are connected by object and data properties. Often, these connections are quite similar or have similar semantics. The *Properties* ontology (see excerpt in Figure 6) defines a common set of object and data properties, which can be reused across the whole ontology. This includes generic properties to model *-to-one and *-to-many relationships like ***has*** and ***hasMultiple***, and specific ones like ***filename*** or ***filetype*** to connect the respective properties to file-like classes and instances.

(cc) BY-SA

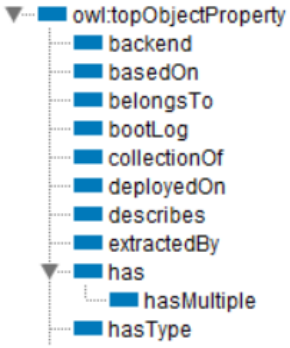

Figure 6. Excerpt of the Properties sub-ontology

2.2.2.1.5 Security – containing Security Feature

Security models security properties for all kind of domains (see Figure 7). Concepts in this ontology include:

- ***Auditing*** – including anomaly detection or logging, for example.
- ***Authenticity*** – including different types of authentications like passwords, OTP or SSO, for example.
- ***Authorization*** – including firewalls or access control, for example.
- ***Availability*** – including backups or redundancy, for example.
- ***Confidentiality*** – including transport encryption or encryption at rest, for example.
- ***Integrity*** – including signatures or hashes, for example.
- ***Reliability*** – including ML-related scores for robustness or explainability, for example.

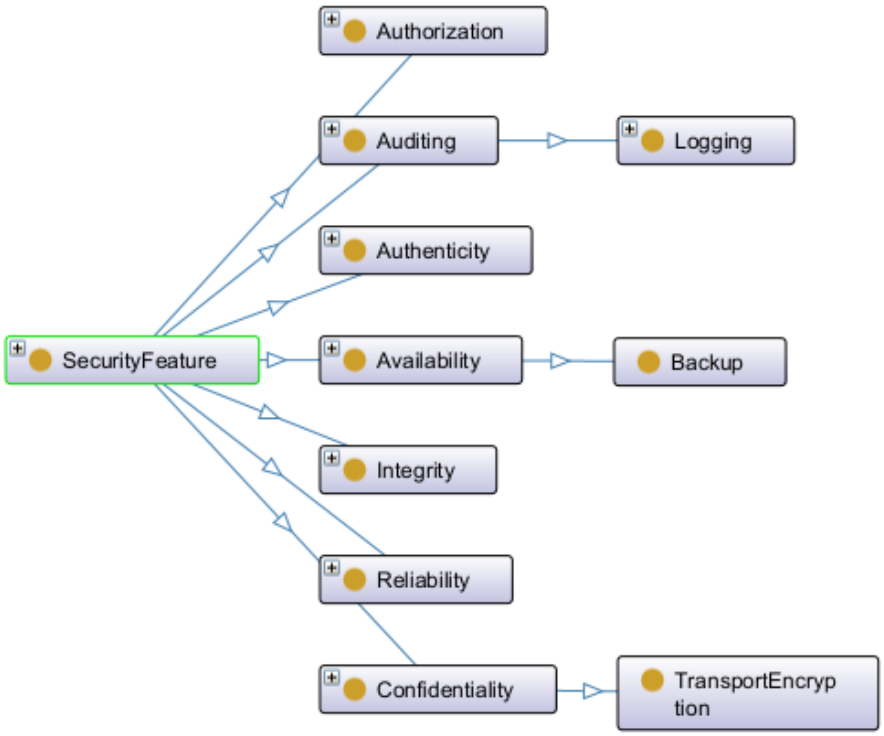

Figure 7. Excerpt of the Security sub-ontology

(cc) BY-SA

2.2.2.2 *Cloud* – an ontology extension for cloud resources

Cloud classifies cloud resources across major cloud providers and architectures, like Microsoft Azure, Amazon Web Services, Google Cloud Platform, OpenStack, and IONOS (see Figure 8). High-level concepts in this ontology are ordered by cloud service categories according to functional purposes and include, among others:

- ***CICDService*** – including jobs, and workflows.
- ***Storage*** – including file and data base storage.
- ***Compute*** – including different types of compute resources like containers, functions, and virtual machines.
- ***Credential*** – including certificates, keys, and secrets.
- ***Networking*** – including virtual networks, and load balancers.

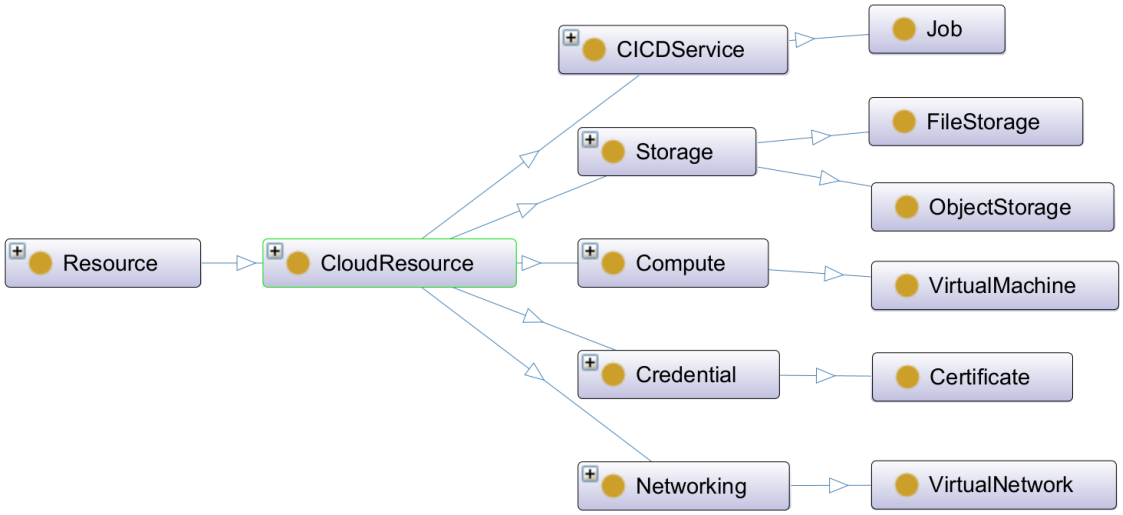

Figure 8. Excerpt of the Cloud ontology extension

2.2.2.3 *Application* – an ontology extension for source code

Application models source code and code-like artifacts (see Figure 9). We base this extension on the general idea of a code property graph, which includes data flow information and can be extended with dedicated nodes and edges, to represent privacy-related properties [12]. This includes extending links to other classes and refining the abstraction level, as just storing the syntax tree would be far too detailed.

High-level concepts in this ontology include:

- ***Component*** – models large software components and forms the base class of *Application* and *Library*.
- ***Module*** – models small software components like source code files.
- ***Application*** – models source software applications and stores properties like the programming language.
- ***Library*** – describes dependencies of components.
- ***SourceCodeFile*** – represents the file containing the source code.

(cc) BY-SA

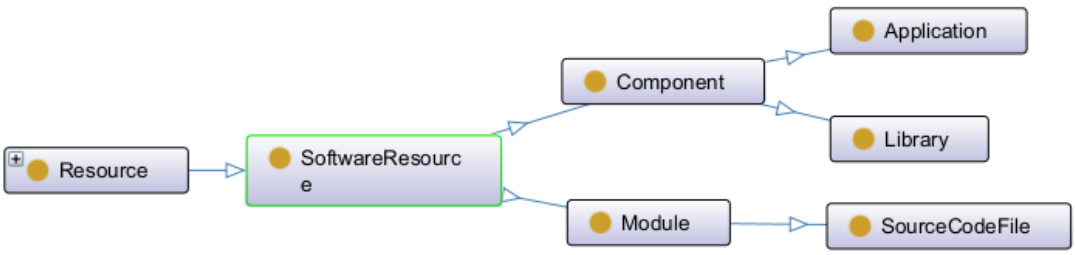

Figure 9. Excerpt of the Application ontology extension

2.2.2.4 *ML* – an ontology extension for AI/ML models

A taxonomy for assessing security-related criteria for ML models deployed in the cloud serves as a structured framework to evaluate, categorize, and mitigate potential threats and security vulnerabilities (see Figure 10). Whereas scientific work (e.g. [14]) provides a comprehensive taxonomy on deep learning techniques, in EMERALD we focus on required data to assess key criteria such as robustness against adversarial attacks and explainability (transparency and interpretability of decisions). We base our work on existing research [15], a query-based method for generating gradient-like explanations without requiring internal model access, to be able to provide a generic approach that can be applied to various types of ML models.

To assess the robustness and explainability scores of a ML model, the following information is required in the taxonomy on a high conceptual level:

- ***MachineLearningModel*** – representing the model information, i.e., the model architecture (including model parameters, hyperparameters, loss function, etc.), the respective task (image recognition, NLP, etc.), the required input and output data types and formats, and the evaluation metrics (such as accuracy, response time, confidence, etc.).
- ***MachineLearningDataset*** – specifying the data used by the model, or its subset.

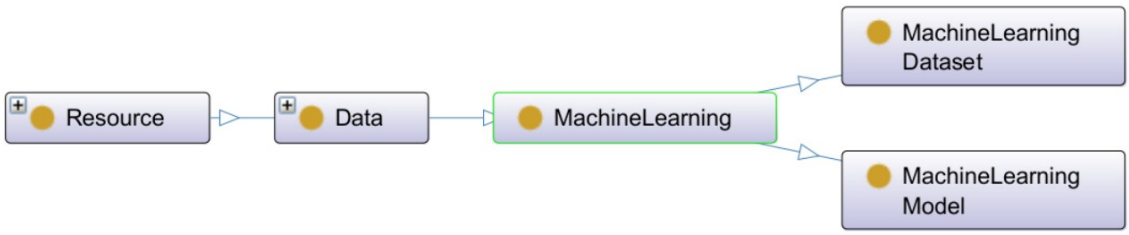

Figure 10. Excerpt of the ML ontology extension

2.2.2.5 *Document* – an ontology extension for security-related documents

Creating a taxonomy for documents, which primarily contains human-readable text (see Figure 11), for automatically assessing security policies and standards involves organizing content into hierarchical or categorized groups that reflect the nature, purpose, and context of the documents [16] [17] [18] [19].

High-level concepts, with many properties attached to cover all required security metrics, in this taxonomy include:

- ***PolicyDocument*** – documenting policies regarding information security, acceptable use, data protection, password, encryption, authentication, etc.
- ***SecurityAdvisoryDocument*** – documenting regulators requirements, internal guidelines, etc.

(cc) BY-SA

- ***ServiceMetadataDocument*** – documenting information on network security, application security, secure software development lifecycle, etc.
- ***GenericDocument*** – representing a placeholder for additional documents that are not yet modelled separately.

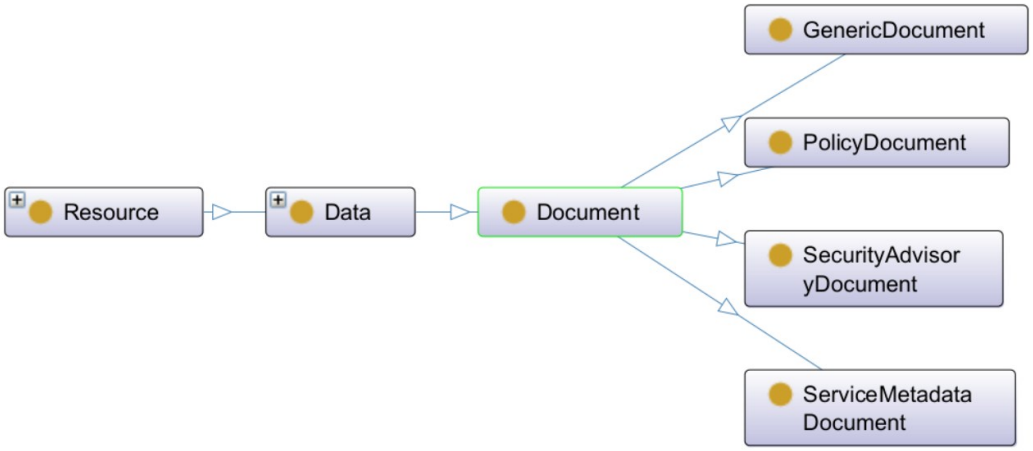

Figure 11. Excerpt of the Document ontology extension

2.2.2.6 *Human Resources* – an ontology extension for trainings

A taxonomy for assessing the status of trainings (such as, an ***AwarenessTraining*** and a ***SecurityTraining***) in an organization is added due to pilot requirements (see Figure 12). This includes the completion progress of trainings and the completion of annual updates.

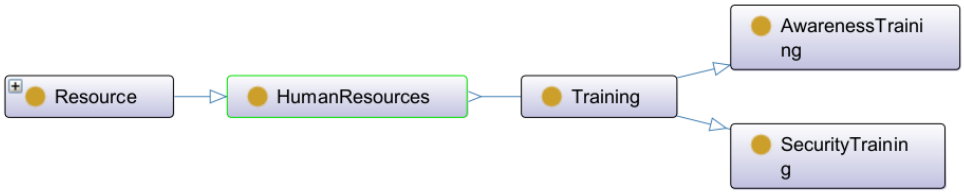

Figure 12. Excerpt of the Human Resources ontology extension

2.2.3 Technical specifications

The *Protégé*[7] and *Git*[8] tools are used to develop the *CertGraph* ontology in EMERALD. *Protégé* is a desktop application developed by Stanford university that enables the modelling of ontologies using OWL concepts. It supports structuring and linking of concepts using different namespaces, i.e., through different sub-ontologies and extensions, by splitting and merging of the ontology into/from multiple files. A reasoning component can derive new information based on rules, which is very useful for the fusion of multiple evidence parts and can detect inconsistencies in ontologies.

All sub-ontologies and extensions are saved as OWL/XML[9]. Changes are checked into the *Git* repository. The discussion and review of these changes occur via pull requests on *GitHub*, before the changes are merged into the main branch. This process ensures that the created ontologies are secured in the sense of allowing for version control, to make sure that the newly developed

[8] https://www.git-scm.com/
[9] https://www.w3.org/TR/owl-xmlsyntax/

(cc) BY-SA

ontologies are discussed, and only corrected and accepted versions are merged into the main branch.

2.3 Delivery and usage

The following sub-sections detail the delivery and usage of the *CertGraph* ontology.

2.3.1 Download

The *CertGraph* ontology is available from the public *Collaborative Metrics Repository* on GitHub[10]. Previously, it was available on a public TECNALIA hosted GitLab repository. The shift enables collaboration and further development across projects within the *European Cluster for Cybersecurity Certification* (EC3)[11] and beyond projects. It also opens contribution possibilities from external entities, which is highly welcomed. The repository hosts all sub-ontologies and extensions in the OWL/XML ontology format (*.owx).

2.3.2 Package information

Table 9 shows the structure of the Github repository[10] and its content.

Table 9. Overview and description of the package structure for the CertGraph ontology

Folder / File	Description
`ontology.owx`	Entry point of the whole ontology, which can be used to open it in *Protégé*, for example (used for convenient development within the EMERALD project)
`core.owx`	*Core* ontology. In addition, it imports the five core ontologies below
`core/evidence.owx`	*Evidence* Ontology
`core/framework.owx`	*Framework* Ontology
`core/functionality.owx`	*Functionality* Ontology
`core/properties.owx`	*Properties* Ontology
`core/security.owx`	*Security Feature Ontology*
`resource.owx`	Wrapper ontology, which imports the five extension ontologies below (used for convenient development within the EMERALD project)
`resource/infrastructure.owx`	Ontology extension for cloud resources
`resource/application.owx`	Ontology extension for source code
`resource/ml.owx`	Ontology extension for machine learning models
`resource/document.owx`	Ontology extension for documents
`resource/human_resources.owx`	Ontology extension for human resources

2.3.3 Instructions for use

Instructions for use are provided as part of the README in the public GitHub repository[12].

In summary, the following requirements must be met before using the *CertGraph* ontology:

- To explore the ontology, an ontology modelling tool that supports the OWL ontology format, such as *Protégé*, must be installed.

[10] https://github.com/Cybersecurity-Certification-Hub/security-metrics/tree/main/ontology
[11] https://cybersecuritycertcluster.eu/
[12] https://github.com/Cybersecurity-Certification-Hub/security-metrics/blob/main/ontology/README.md

(cc) BY-SA

- If the ontology should be extended or changed, a version control tool, such as Git, is recommended.

The whole ontology can be viewed in *Protégé* by opening `ontology.owx`. In addition, if only parts of the ontology are to be viewed, the respective owx file can also be opened on its own.

To instantiate the ontology, the following workflow is recommended:

1. Create a new ontology file.
2. Import `core.owx` using the *Imported Ontologies* view.
3. Import relevant extensions from the resource folder or all of them by importing `resource.owx`.
4. Created instances will be stored in the newly created ontology file.

Note that the *CertGraph* ontology will not be visualized in the EMERALD UI.

2.3.4 Licensing information

The *CertGraph* ontology and its sub-ontologies and extensions are licensed as open source under Apache License, Version 2.0.

2.4 Future development of the ontology

The *CertGraph* ontology with its sub-ontologies and extensions will continuously be extended during the project and beyond. The shift to the *Collaborative Metrics Repository* on GitHub opens new possibilities for the extension of the ontology also across projects.

Due to the continuous development of metrics and new application domains according to evolving requirements, future changes in the ontology are expected. This mainly includes new concepts, properties connecting these concepts to existing ones, or properties for additional values. However, the overall structure of the ontology (i.e., *Core* ontologies and the classification of concepts into extensions) is expected to remain unchanged.

DRAFT

(cc) BY-SA

3 Refined illustrative example of modelling and combining evidence information for the used TLS Version

For better illustration, we base the idea for modelling and combining evidence information from different sources on an example (see Figure 13), which uses a selected security criteria "Encryption of data for transmission" that is specified in the BSI C5:20201 (CRY-02). This control requires that a cloud service provider has established procedures and technical measures to ensure strong encryption and authentication of data in transit. This means that customer data must be protected by state-of-the-art encryption mechanisms (e.g., TLS (*Transport Layer Security*) with strong cipher suites) when it is transmitted over public networks, ensuring the confidentiality and integrity of the data while it is moving between systems or services. In this example we model the used TLS version from different views.

This is a refinement of the first idea drafted in D2.1 [1]. There were no conceptual changes from D2.10 [2]. The focus is on illustrating the interconnectivity between selected sub-ontologies and extensions, and not on contained details or reflecting the whole hierarchy. The restructuring and extension of the *CertGraph* ontology is reflected in Figure 13. In the diagram, classes are visualized as rectangles and instances as hexagons. Open-headed arrows with a solid line (→) represent "subclass of" relations, which connect subclasses to their parent class, and open headed arrows with a dashed line (⇢) represent "instance of" relations, which connect instances to their class. Simple arrows (→) represent data and object properties. These arrows are used between classes to define the schema, as well as between instances in their materialized form.

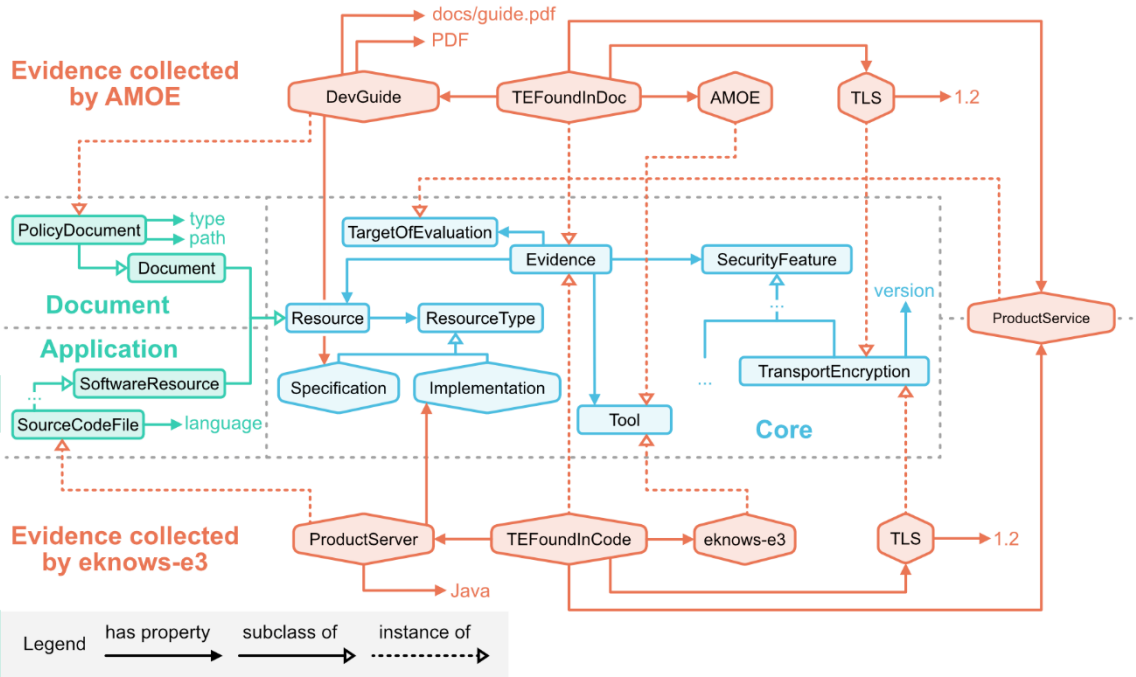

Figure 13. Classes (rectangles) and instances (hexagons) for the TLS example, showing evidence found in source code (implemented) and corresponding evidence in a document (specified) regarding transport encryption, which can be used to verify CRY-02 from BSI C5:2020 (adapted from [9])

As described in Section 2.2.2, the ontology ***Core*** forms the basis for the *CertGraph* ontology. It defines the metamodel for EMERALD evidence and uses the concepts defined in the ***Security*** sub-ontology, which contains a variety of security features and data properties:

(cc) BY-SA

- ***Evidence*** is the central class and instances of it represent detected security evidence. Each evidence has connections to *SecurityFeature, TargetOfEvaluation, Resource,* and *Tool*.
- ***Resource*** represents the source of a piece of evidence and stores relevant metadata for the location. Each *Resource* has a connection to an *ResourceType*.
- ***ResourceType*** classifies the role of resource within the system. *ResourceType* is modelled as an enumeration type in ontology terms. For this, a class is needed, and an instance is created for each possible variant. Currently, we distinguish between these two possible variants:
    - The first variant, ***Specification***, is used for evidence found in resources, which describe how the system *should* behave. The main application for this variant is in human-readable documents which are not automatically processed for compilation, e.g. policy documents.
    - The second variant, ***Implementation***, is used for evidence found in resources, which describe, how the system *actually* behaves. This variant is mainly used for evidence found in machine-processed assets, e.g. source code, configuration files, or runtime information.
- ***TargetOfEvaluation*** ties the evidence to a certain service. This connection enables the fusion of evidence from multiple sources using a unique identifier for each service, which will be used as URI for the service instance.
- ***Tool*** represents the extractor component that has collected the evidence.
- To keep things simple, only a single feature (***TransportEncryption*** class) is showcased in this example, and the hierarchy has also been simplified to two levels. The base class of this hierarchy is ***SecurityFeature***. Also, for simplicity reasons, just one data property ***version*** is shown to store the TLS version.

Resource (defined in *Core*) is the starting point for ontology extensions. In this example, we used the ***Document*** and ***Application*** extensions, which are built on top of the *Core* ontology, and limit the scope to just one class per extension. As previously described, the classes in the extensions should model their respective domains. The following two classes are used in the example:

- ***PolicyDocument*** represents a human-readable textual document for policies. It is modelled as a sub-class of ***Document*** and includes (*has*) two shown data properties ***type*** and ***path***.
- ***SourceCodeFile*** represents a source code file which is compiled for a given service and is stored in a repository. It is modelled as a (indirect) sub-class of ***SoftwareResource*** and includes (*has*) a data properties ***language.***

Gathered evidence provided by tools is modelled by instantiating classes defined in *Core* and in extensions. In the example in Figure 13, evidence for the target of evaluation ***ProductService*** is provided by two extraction components:

- ***AMOE*** scanned the ***DevGuide*** (a PDF document stored at *docs/guide.pdf*) and found that ***TLS*** version *1.2* is required to be used in development.
- ***eknows-e3*** scanned the ***ProductServer*** (written in *Java*) and found that ***TLS*** version *1.2* is used in the implementation.
- Found evidence is represented as the instances ***TEFoundInDoc*** and ***TEFoundInCode***, which have respective connections to the other instances. Please note that "TransportEncryption" is abbreviated as "TE" in the diagram.

To sum up, the example illustrates the key idea of the *Certification Graph* to represent security-related parts of a cloud service, e.g. of the source code, in a graph structure and provides

(cc) BY-SA

additional context through the discovery of other related cloud resources, e.g. policy documents. Bridging different domains allows to combine evidence at a higher level of knowledge and enables a comparison, for example, of what is described in policy documents and what is actually implemented in software. An important point for maximising the potential of the *Certification Graph* is that evidence from other extraction components must link to the same service instance. In OWL, two instances are considered as *the same* if they are identified by the same URI. This enables knowledge fusion for the assessment in WP3.

DRAFT

(cc) BY-SA

4 AIOX – AI-assisted Ontology Extension Tool for CertGraph Refinement

4.1 Functional description

Motivation and scope. While the modular and extensible design of the *CertGraph* ontology allows continuous evolution, its practical maintenance and extension remain a labour-intensive and error-prone task. Metric definitions[13], expressed in human-readable YAML files, represent an important intermediate artefact between security standards and automated evidence assessment. However, the semantic gap between these YAML-based definitions and the formal OWL ontology requires substantial expert effort. In particular, the introduction of new security metrics, derived from evolving standards, security controls, or project-specific assessment needs, frequently reveals previously uncovered concepts, properties, or specialisations that must be consistently integrated into the ontology.

Manually identifying these new concepts, determining their correct placement within the existing ontology hierarchy, and modelling their properties in OWL requires substantial domain expertise and deep familiarity with the ontology structure. This process does not scale well with the growing number of metrics and domains addressed within the EMERALD project and beyond. To address this challenge, the *AI-assisted Ontology Extension* (AIOX) tool has been developed to support ontology engineers and domain experts by semi-automating the identification, suggestion, and integration of new ontology elements. The tool leverages AI-based semantic analysis to reduce manual effort while ensuring that all changes remain under expert control.

Functional overview. The AIOX tool is an interactive command-line utility that supports domain experts in extending the *CertGraph* ontology based on information extracted from YAML metric definitions. These metric files act as an intermediate representation between security standards and the formal ontology.

The tool performs the following core functions:

- *Concept extraction from metrics*: It parses YAML metric files and identifies candidate ontology terms, such as classes and properties, based on structured patterns in metric descriptions and configurations.
- *Detection of ontology gaps*: Extracted terms are compared against the current ontology. Existing terms are flagged for consistency checks, while unknown terms are treated as candidates for ontology extension.
- *AI-assisted placement of new classes*: For each new class candidate, the tool suggests suitable parent classes within the existing ontology hierarchy and guides the user through the final placement decision.
- *Property identification and linking*: The tool proposes new data properties derived from metric configurations, infers their data types, and suggests the most appropriate class to which the property should be attached.
- *Interactive expert validation*: All suggested additions are presented to the user through interactive prompts, allowing confirmation, rejection, or manual adjustment before any modification is applied.
- *Controlled ontology modification*: Upon confirmation, the tool updates the OWL ontology and automatically creates a backup of the original file to ensure traceability and rollback capability.

[13] https://github.com/Cybersecurity-Certification-Hub/security-metrics/tree/main/metrics

(cc) BY-SA

The tool is explicitly designed as a semi-automated assistant, not a fully autonomous ontology generator. Human validation remains an integral part of the workflow.

4.2 Technical description

The following subsections describe the technical details of the AIOX tool.

4.2.1 Architecture

The tool is implemented as a standalone Python command-line application *(ontology_extender.py*) and operates locally on the *CertGraph* OWL files. Its architecture follows a modular pipeline (see Figure 14) consisting of:

1. The *ontology parsing module*, which parses the existing *CertGraph* ontology to extract existing classes, properties, hierarchies, and annotations. This internal representation is used as the baseline for comparison and extension.
2. The *metric analysis module,* which processes YAML metric files, extracting structured concepts enclosed in square brackets from natural-language descriptions, as well as configuration parameters such as properties and target values.
3. The *AI-assisted recommendation module*, which applies NLP and embedding-based methods to generate ranked suggestions for class placement and property attachment.
4. An *interactive decision layer, which* guides the user through a sequence of prompts to validate or override AI-generated recommendations.
5. The *ontology update module,* which applies approved changes to the ontology, while automatically creating a backup of the original OWL files.

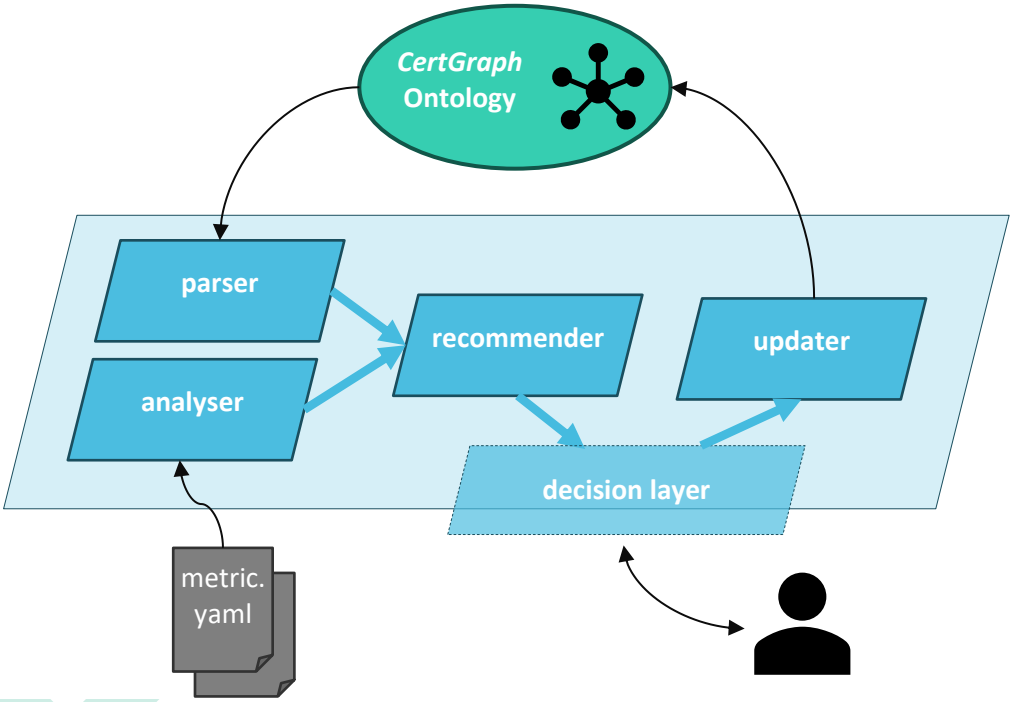

Figure 14. Modular pipeline of the AIOX tool

4.2.2 Technical specifications

The AIOX tool uses the following methods and techniques:

- *Semantic similarity with sentence transformers*: To suggest suitable parent classes for new concepts, the tool uses a sentence transformer model (*all-MiniLM-L6-v2*) to

(cc) BY-SA

compute semantic similarity between new candidate concepts and existing ontology classes. To improve recommendation quality, the tool implements an aggregated embedding strategy, where a parent class is represented not only by its own label but also by the embeddings of its direct subclasses. This approach reflects the intuition that semantic similarity is often strongest among sibling concepts rather than direct parents.

- *Property recommendation via masked language models (MLM)*: Linking a newly identified property to the correct class proved to be a more complex challenge. After evaluating several approaches, the tool adopts a data-type-aware MLM-based strategy. The tool infers the property's data type (e.g., Boolean, string) from the metric configuration. Based on the inferred type, it constructs a structured sentence template with a masked class position. A masked language model then evaluates which candidate class best fits the sentence context. The tool also employs a hybrid strategy to balance semantic expressiveness with computational efficiency and includes an automatic fallback to a baseline semantic similarity method if the MLM confidence is low.

4.3 Delivery and usage

The following sub-sections detail the delivery and usage of the AIOX tool.

4.3.1 Download

The AIOX tool is delivered as part of the EMERALD project's tooling repository[14] and can be executed locally by ontology engineers or domain experts. It is intended for use during ontology maintenance phases, particularly when new security metrics are introduced, or existing certification schemes are extended.

The tool only requires standard Python dependencies and access to the *CertGraph* OWL file and metric definitions. Ensure that you have the required Python packages installed:

```
pip install -r extractor/requirements.txt
```

4.3.2 Instructions for use

The tool can be executed either by selecting a metric interactively or by specifying a metric file directly via the command line. During execution, users are guided through a sequence of prompts that support informed modelling decisions. Before any changes are applied, the tool presents a comprehensive summary of the proposed modifications to the ontology for final confirmation.

Example. To illustrate the usage of the AIOX tool, we again consider the example on the TLS version, i.e. the metric *TransportEncryptionEnabled*, defined in a YAML file (see Figure 15).

```yaml
# ====== Metadata ======
id: TransportEncryptionEnabled
description: This rule assesses whether a [Resource] has [TransportEncryption] [p1:enabled] correctly configured.
category: TransportEncryption
version: "1.0"
comments: Transport encryption is a standard practice to keep sensitive data confidential when it is transmitted.
# ====== Configuration ======
configuration:
  p1:
    operator: "=="
    targetValue: True
```

Figure 15. Example metric: TransportEncryptionEnabled.yaml

[14] https://git.code.tecnalia.dev/emerald/public/components/ontext/ontext

(cc) BY-SA

To run the tool with a specific metric file, use the following instruction:

```
python extractor/ontology_extender.py
metrics/TransportEncryption/TransportEncryptionEnabled/TransportEncrypti
onEnabled.yaml
```

During the analysis of the metric file, the tool extracts the following candidate ontology elements from the metric description and configuration sections:

- Class candidates: `Resource`, `TransportEncryption`
- Property candidate: `enabled`
- Inferred property type: `xsd:boolean` (derived from the target value `True`)

While `Resource` is already present, `TransportEncryption` is identified as a new class requiring integration. Using AI-based semantic similarity, the tool suggests suitable parent classes and presents them to the user in an interactive prompt, including hierarchical context (see Figure 16).

```
Found new term: [TransportEncryption]

Suggested top 3 parent classes:

1: AtRestEncryption (SecurityFeature → Confidentiality →
AtRestEncryption)

2: Confidentiality (SecurityFeature → Confidentiality)

3: EncryptionOperation (Functionality → Operation →
CryptographicOperation → EncryptionOperation)

Select parent class (1–3), enter manually, or skip (0):
```

Figure 16. Interactive prompt: New term identification

After user confirmation, the tool proceeds to the analysis of the properties. It recommends adding a Boolean property `enabled` and evaluates to which class the property should be attached. Using an MLM, it compares alternatives such as "Resource is enabled" and "TransportEncryption is enabled" and selects the more semantically plausible option. The user is prompted accordingly (see Figure 17).

```
Suggested property: enabled (xsd:boolean)

Recommended class: TransportEncryption

(Similarity: 0.92)(

Add property to class? (y/n):
```

Figure 17. Interactive prompt: Property suggestion

Please also refer to the README.md file[15] which contains further instructions on setup and usage of the AIOX tool.

4.3.3 Licensing information

The AIOX tool is licensed as open source under Apache License, Version 2.0.

[15] https://git.code.tecnalia.dev/emerald/public/components/ontext/ontext

(cc) BY-SA

4.4 Future developments

While the current tool focuses on class and data property integration, several extensions are considered beyond the scope of this project. These include fully automated extraction of ontology concepts directly from security standards, advanced detection of object properties representing relations between classes, and the introduction of quantitative evaluation mechanisms to assess recommendation quality. These directions provide a clear pathway for future research and tool evolution but are not addressed within the current project scope.

DRAFT

(cc) BY-SA

5 Conclusions

In this deliverable, we described the final version of the *CertGraph* ontology, which is the central schema for integrating evidence extracted from multiple cloud service layers (i.e., infrastructure, data, platform, and software), including source code, policy documents, ML models, and runtime information, into a single graph-based structure (KR2-CERTGRAPH).

The main purpose of this ontology is to harmonize evidence gathering and assessment. Security controls defined in different certifications or catalogues can be assigned to ontological concepts and those ontological types can be further used in metric definitions. Therefore, the ontology defines a vocabulary for mapping between the properties that shall be measured and the respective gathering of adequate evidence.

Based on this general idea of (harmonized) security metrics, the *CertGraph* ontology allows different evidence extraction tools to gather and combine different kinds of evidence for the same metric, enhancing reuse of evidence collected, and providing answers to assess the metrics. We provided functional and technical descriptions of the ontology, including its sub-ontologies and extensions to support the holistic approach to evidence collection. These include security features to assess security metrics, as well as different domain concepts and relationships for extracting evidence from source code, from policy documents, from ML models, and from cloud runtime environments. A new addition to these key extensions is the taxonomy for modelling concepts for human relations, such as trainings. We further presented updated instructions for its delivery and usage and discussed directions for future development of the *CertGraph* ontology beyond the EMERALD project and across various projects.

An example of modelling and combining evidence information for TLS encryption (specified in the BSI C5:20201 (CRY-02)) was refined and presented to illustrate the purpose and innovation of the ontology. Finally, an AI-assisted ontology extension tool that supports expert-controlled refinement of the *CertGraph* ontology from new metric specifications was described. In future work, we plan to further modularize the ontology and strengthen its interoperability with other security and compliance standards to facilitate broader reuse and integration.

DRAFT

(cc) BY-SA

6 References

[1] EMERALD Consortium, “D2.1 Graph Ontology for Evidence Storage,” 2024.

[2] EMERALD Consortium, “D2.10 Certification Graph – v1,” 2025.

[3] EMERALD Consortium, “D3.4 Evidence Assessment and Certification-Implementation - v2,” 2025.

[4] EMERALD Consortium, “D2.3 Source Evidence Extractor – v2,” 2025.

[5] EMERALD Consortium, “D2.5 AMOE – v2,” 2025.

[6] EMERALD Consortium, “D2.7 ML model certification – v2,” 2025.

[7] EMERALD Consortium, “D2.9 Runtime evidence extractor – v2,” 2025.

[8] C. Banse, I. Kunz, A. Schneider and K. Weiss, “Cloud property graph: Connecting cloud security assessments with static code analysis,” in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD) (pp. 13-19)*, 2021.

[9] S. Schöberl, C. Banse, V. Geist, I. Kunz and M. Pinzger, “CertGraph: Towards a Comprehensive Knowledge Graph for Cloud Security Certifications,” in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (pp. 76-77)*, 2024.

[10] L. Orue-Echevarria, J. Alonso and et al., “MEDINA: Improving Cloud Services trustworthiness through continuous audit-based certification,” in *CEUR Workshop Proceedings*, 2021.

[11] C. Banse, I. Kunz, N. Haas and A. Schneider, “A Semantic Evidence-based Approach to Continuous Cloud Service Certification,” in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (pp. 24–33)*, 2023.

[12] I. Kunz, K. Weiss, A. Schneider and C. Banse, “Privacy property graph: Towards automated privacy threat modeling via static graph-based analysis,” in *Proceedings on Privacy Enhancing Technologies*, 2023.

[13] EMERALD Consortium, “D1.2 Data modelling and interaction mechanisms - v2,” 2025.

[14] I. Sarker, “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions,” *SN computer science,* vol. 2, no. 6, p. 420, 2021.

[15] Y. Cai and G. Wunder, "On Gradient-like Explanation under a Black-box Setting: When Black-box Explanations Become as Good as White-box. arXiv preprint arXiv:2308.09381," 2023.

[16] J. H. Eloff, R. Holbein and S. Teufel, “Security classification for documents,” *Computers & Security,* vol. 15, no. 1, pp. 55-71, 1996.

(cc) BY-SA

[17] S. W. Lee, R. Gandhi, D. Muthurajan, D. Yavagal and G. J. Ahn, "Building problem domain ontology from security requirements in regulatory documents," in *Proceedings of the 2006 international workshop on Software engineering for secure systems, pp. 43-50*, 2006.

[18] U. Garain and B. Halder, "Machine authentication of security documents," in *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, pp. 718-722*, 2009.

[19] G. Liu and H. Zhang, "An ontology constructing technology oriented on massive social security policy documents," *Cognitive Systems Research,* vol. 60, pp. 97-105, 2020.

DRAFT

(cc) BY-SA