CertGraph: Towards a Comprehensive Knowledge Graph for Cloud Security Certifications

Stefan Schöberl
Software Competence Center Hagenberg GmbH
Hagenberg im Mühlkreis, Austria
stefan.schoeberl@scch.at

Christian Banse
Fraunhofer AISEC
Garching bei München, Germany
christian.banse@aisec.fraunhofer.de

Verena Geist
Software Competence Center Hagenberg GmbH
Hagenberg im Mühlkreis, Austria
verena.geist@scch.at

Immanuel Kunz
Fraunhofer AISEC
Garching bei München, Germany
immanuel.kunz@aisec.fraunhofer.de

Martin Pinzger
University of Klagenfurt
Klagenfurt, Austria
martin.pinzger@aau.at

Abstract

This paper introduces *CertGraph*, a knowledge graph-based approach designed to streamline security certification which integrates evidence from multiple sources. Unlike existing approaches, we consider the complete stack from software to policies, and enable the fusion of evidence from different views and sources. Its extensible ontology is designed to accommodate multiple domains, including cloud security, AI models, and source code. By providing an automated and systematic approach to build an ontology, *CertGraph* aims to facilitate more effective security certification and compliance verification.

ACM Reference Format:
Stefan Schöberl, Christian Banse, Verena Geist, Immanuel Kunz, and Martin Pinzger. 2024. CertGraph: Towards a Comprehensive Knowledge Graph for Cloud Security Certifications. In *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), September 22–27, 2024, Linz, Austria.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3652620.3687795

1 Introduction

Using semantic representations in the field of security certifications has recently gained traction, especially through the MEDINA project [7] and related research in this field [1, 2, 6]. These existing approaches build on the notion of gathering so called *evidence* – from sources such as the cloud infrastructure – to demonstrate compliance to certain standards or regulations. To harmonize evidence gathered from various cloud providers and technologies, a mapping to a structure described in an ontology is performed. However, these previous approaches have several shortcomings. They are not very comprehensive in terms of semantic modelling, for example focusing mostly 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, previous approaches created different, independent kinds of evidence for each resource and stored them into information silos, even if they describe the same aspect (e.g., configuration of encryption), but from different viewpoints.

Therefore, we introduce *CertGraph*, which aims at two aspects. First, CertGraph 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. Furthermore, it provides an initial approach for the *fusion* of evidence coming from different views/sources of the same resource.

2 Related work

There exist several related works in the individual domains that are to be considered in our ontology. Related to cloud security, Joshi et al. [5] proposed a knowledge graph schema for cloud compliance automation. Sikeridis et al. proposed a taxonomy of public cloud vendors [9]. Hendre and Joshi created a taxonomy of security controls and security related standards [4]. While these works contain the general relationship between stakeholders and regulatory statements, they lack specific structure for evidence that need to be gathered. With regards to semantic modelling of AI, Testi et al. [10] provide a systematic overview on MLOps, but not on properties of an AI model itself. Sarker [8] provides a comprehensive overview on the taxonomy of deep learning techniques. They classify techniques in general categories such as supervised vs. unsupervised learning and describe further properties of these techniques. Finally, approaches such as code property graphs [11, 12] focus on semantic abstraction of source code from different programming languages [3] but lack higher-level concepts.

3 Building the Knowledge Graph

The foundation of our knowledge graph is an ontology and the fusion of knowledge from different sources. For better illustration, we base the explanations in this section on an example (see Figure 1), which uses one selected security criteria: *Encryption of data for transmission*, which is specified in the BSI C5:2020[1] (CRY-02). In this example we model the used TLS (Transport Layer Security) version from different views.

Funded by the EU project EMERALD, Grant No. 101120688.

MODELS Companion '24, September 22–27, 2024, Linz, Austria

ACM ISBN 979-8-4007-0622-6/24/09
https://doi.org/10.1145/3652620.3687795

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

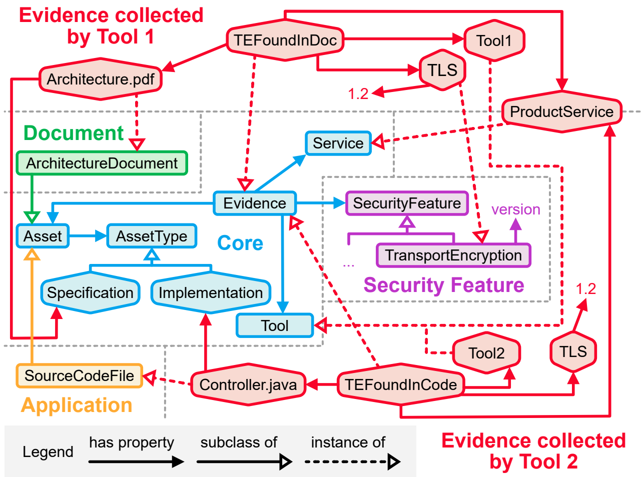

Figure 1: Classes (rectangles) and instances (hexagons) for the TLS example, showing an evidence found in source code (implemented) and a corresponding evidence in an architecture document (specified) regarding transport encryption, which can be used to verify CRY-02 from BSI C5:2020.

3.1 Ontology design

We propose an ontology to store and link evidence, which is automatically extracted from different sources.

The *CertGraph Ontology* consists of multiple smaller ontologies. As shown in Figure 1, two ontologies form the base: *Core* and *Security Feature*. *Security Feature* models security properties and is based on the taxonomy with the same name from the Cloud Property Graph [2]. *Core* 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 an *Asset* (to store the detection point for traceability) and to a *Service* (to link to the related cloud service). *Asset* has a connection to *AssetType* (modeled as an enumeration type), to distinguish between specified and implemented behavior.

Extensions are built on top of *Core* and hook into the *Asset* taxonomy. We propose four extensions, each covering its own domain:

Document to model policy and organizational documents, which primarily contain human-readable text, like the *ArchitectureDocument* in Figure 1.

Application to model source code and code-like artifacts. Here a suitable abstraction level has to be found, which focuses on links to other components, usage of libraries, and operations. An initial approach has been described by Kunz et al. [6]. Just storing the syntax tree would be far too detailed. Figure 1 illustrates this with the *SourceCodeFile*, which refers to a single file as a whole. Additional properties, like line and column numbers could be added.

Cloud to model cloud resources and this extension is based on the *CloudResource* taxonomy from the Cloud Property Graph [2].

ML to model machine learning models deployed in the cloud. A suitable starting point could be the Deep Learning taxonomy described by Sarker [8]. More details (properties, etc.) need to be extracted from the textual description of each technique.

This approach also allows for further extension of the ontology by developing new extensions for other domains, if needed.

3.2 Approaches for knowledge fusion

To meaningfully fuse the knowledge, which is provided by the evidence extraction tools, we propose a variety of ideas on how to accomplish this. One idea is to use SWRL[2] or similar languages to describe rules, which are used to derive new knowledge from gathered evidence, thus new edges are added to the graph, which in turn leads denser interlinking of data. In this context, it has already became apparent that a unique ID is probably necessary to identify service instances (i.e., each service can be referenced by a unique URI across extractors). Another idea is to use SPARQL[3] to query the graph and in this way to link the information in the graph and receive it as a query result. Currently, we are evaluating, what can be implemented, which libraries are available, and what is supported by the used graph database.

4 Outlook

Next steps include further formalization of concepts like the ML extension. We are also looking for collaborations with other domains that can be included in the ontology as well. Furthermore, the fusion of knowledge has to be modeled and implemented in software, whereby it must be evaluated in advance, which formalism is supported by libraries and databases.

References

[1] Christian Banse, Immanuel Kunz, Nico Haas, and Angelika Schneider. 2023. A Semantic Evidence-based Approach to Continuous Cloud Service Certification. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*. 24–33.
[2] Christian Banse, Immanuel Kunz, Angelika Schneider, and Konrad Weiss. 2021. Cloud property graph: Connecting cloud security assessments with static code analysis. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*.
[3] Verena Geist, Michael Moser, Josef Pichler, and Florian Schnitzhofer. 2024. Innovating Industry with Research: eknows and Sysparency. *IEEE Software* (2024).
[4] Amit Hendre and Karuna Pande Joshi. 2015. A semantic approach to cloud security and compliance. In *2015 IEEE 8th International Conference on Cloud Computing*. IEEE, 1081–1084.
[5] Karuna Pande Joshi, Lavanya Elluri, and Ankur Nagar. 2020. An integrated knowledge graph to automate cloud data compliance. *IEEE Access* 8 (2020), 148541–148555.
[6] Immanuel Kunz, Konrad Weiss, Angelika Schneider, and Christian Banse. 2023. Privacy property graph: Towards automated privacy threat modeling via static graph-based analysis. *Proceedings on Privacy Enhancing Technologies* (2023).
[7] Leire Orue-Echevarria, Juncal Alonso, et al. 2021. MEDINA: Improving Cloud Services trustworthiness through continuous audit-based certification. In *CEUR Workshop Proceedings*. CEUR-WS.
[8] Iqbal H Sarker. 2021. Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN computer science* 2, 6 (2021).
[9] Dimitrios Sikeridis, Ioannis Papapanagiotou, Bhaskar Prasad Rimal, and Michael Devetsikiotis. 2017. A Comparative taxonomy and survey of public cloud infrastructure vendors. *arXiv preprint arXiv:1710.01476* (2017).
[10] Matteo Testi, Matteo Ballabio, Emanuele Frontoni, Giulio Iannello, Sara Moccia, Paolo Soda, and Gennaro Vessio. 2022. MLOps: A Taxonomy and a Methodology. *IEEE Access* 10 (2022), 63606–63618.
[11] Konrad Weiss and Christian Banse. 2022. A language-independent analysis platform for source code. *arXiv preprint arXiv:2203.08424* (2022).
[12] Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. 2014. Modeling and discovering vulnerabilities with code property graphs. In *2014 IEEE symposium on security and privacy*. IEEE, 590–604.

[2]https://www.w3.org/submissions/SWRL/
[3]https://www.w3.org/TR/sparql11-query/