Modelling Tool Extension for Vulnerability Management

Avi Shaked, Nan Zhang Messe, Tom Melham

Modelling Tool Extension for Vulnerability Management

Avi Shaked[†]
Department of Computer Science
University of Oxford
Oxford, UK
avi.shaked@cs.ox.ac.uk

Nan Messe
IRIT
UT2
Toulouse, France
nan.messe@irit.fr

Tom Melham
Department of Computer Science
University of Oxford
Oxford, UK
tom.melham@cs.ox.ac.uk

ABSTRACT

Managing vulnerabilities with respect to the design of systems is essential to securing systems and establishing their trustworthiness. Until now, there has been no modelling tool to support vulnerability management within the context of system design. We present a new, open-source extension of a systems security design and assessment tool. First and foremost, this extension integrates a pertinent vulnerability management domain ontology into the tool's underlying metamodel. Based on the extended metamodel, the enriched tool supports importing information from vulnerability-related knowledge bases as well as capturing new vulnerability information and security rules. This information can then be used in an integrative and scalable form to analyse and reason about the security of systems designs. The extended tool now includes an automated reasoning mechanism for establishing the vulnerability posture of systems designs.

CCS CONCEPTS

• Computing methodologies→ Modeling and simulation→ Model development and analysis→ Modeling methodologies • Security and privacy→ Systems security→ Vulnerability management • Security and privacy→ Software and application security→ Software security engineering

KEYWORDS

Model driven engineering, Threat modelling, Vulnerability management, Security by design

ACM Reference format:

Avi Shaked, Nan Messe and Tom Melham. 2024. Modelling Tool Extension for Vulnerability Management. 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, 5. https://doi.org/10.1145/3652620.3687791

[†]Corresponding author.

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

ACM ISBN 979-8-4007-0622-6/24/09.
https://doi.org/10.1145/3652620.3687791

1 Introduction

Establishing and maintaining the vulnerability posture of a system is critical to securing the system and assessing its trustworthiness. These efforts are commonly known as vulnerability management. Vulnerability management allows involved stakeholders (e.g., designers, risk managers, and executives) to identify, understand and mitigate potential security risks, consequently improving the trustworthiness of systems [5, 6, 13].

Vulnerability management practices are typically reactive, and are traditionally approached with a "patching" state of mind, i.e., disclosed vulnerabilities in system components are mitigated by applying software updates (patches). This is well exemplified by the recent UK National Cyber Security Centre guidance on vulnerability management [13]. A more mature mindset is to secure systems by design, eliminating entire classes of vulnerabilities by proper design and incorporation of security controls [2]. Vulnerability classes are often referred to as weaknesses.

The security posture of systems is constantly evolving. This is due to two main reasons: 1) new functionality and updates are introduced into systems, either during initial development or throughout the operational lifecycle; 2) new vulnerabilities are disclosed, and – as their exploits become widely available – attackers' capabilities increase. These necessitate an ongoing vulnerability management effort, which should be underpinned by a computer-aided tool. Previous works offer tools for vulnerability management. Recent examples include the Cyber Intelligence Alarm system, which alerts about vulnerabilities and countermeasures[10] ; Vulnerability Management Center, which scores vulnerabilities of organisational assets[14]; and AMADEUS-Exploit, which generates feature models of vulnerabilities[11]. They are focused on implementation-level vulnerabilities, and none of these address vulnerability classes and mitigation planning, which are essential to security by design. Furthermore, to our knowledge, there is no modelling tool available to support vulnerability management in context of system design, let alone a tool that is driven by a rigorous, research-informed metamodel (as in a model-driven engineering approach).

In this article, we present a new extension of an existing security modelling tool. The extension introduces – for the first

time – concepts related to vulnerability management into the modelling tool's underlying metamodel, and provides integrated metamodel-driven vulnerability management functionality. In Section 2, we provide background about the modelling tool that we used as an infrastructure as well as on pertinent knowledge bases that the extended tool uses. In Section 3, we describe the new vulnerability management functionality and detail how it is implemented in the newly extended tool. Finally, in Section 4, we conclude by highlighting the benefits and potential of the extended tool.

2 Background

TRADES Tool [8] is an open-source systems security design and assessment tool. It is a domain-specific modelling tool which relies heavily on a domain metamodel and carefully designed representations. TRADES Tool is based on Eclipse infrastructure, and utilises Ecore for its metamodel and Sirius for its metamodel-driven representations. TRADES Tool also supports the integration of several, widely used security knowledge bases [9]. However, until now, TRADES Tool was limited to a threat-oriented security design and analysis perspective, based on ontological concepts such as *threat pattern*, *allocated threat / risk*, *component* and *security control*. Specifically, it did not offer a vulnerability-oriented security design and analysis perspective.

In our research work to extend TRADES Tool to support vulnerability management, we derived the pertinent domain ontology, i.e., concepts and relations that characterise the domain. While the full details of uncovering the vulnerability management domain concepts and designing the ontology extend beyond the scope of this article, we highlight two knowledge bases that were used for this: CWE (Common Weakness Enumeration) and CVE (Common Vulnerabilities and Exposures).

The standard security terminology by MITRE – a global leader in systems engineering and cyber security – considers *weaknesses* as classes of vulnerabilities at various levels of abstraction, and *vulnerabilities* as implementation vulnerabilities. Implementation vulnerabilities can manifest weaknesses [3, 4]. The online CWE knowledge base is a hierarchical organisation of weaknesses [1]. The CWE knowledge base offers its content as a downloadable archive of structured information. Since this knowledge base offers class-related definitions, the content is not frequently updated and is relatively manageable (as of July 2024, a total of 938 weaknesses). The online CVE knowledge base holds details about implementation vulnerabilities [7]. New CVEs are being disclosed daily, and it is therefore unrealistic to use a single archive download. NIST provides an Application Programming Interface (API) for querying its online CVE knowledge base (the National Vulnerability Database, NVD)[12].

3 Tool Extension for Vulnerability Management

The TRADES Tool extension for supporting vulnerability management uses the same infrastructure of the original tool (namely, Eclipse and its different features). The extension is an inherent part of the main tool, with the new functionality fully integrated into the previously-implemented functionality. It is available under a permissive EPL-2.0 license at https://github.com/UKRI-DSbD/TRADES.

Figure 1 shows the elements of the extended tool's metamodel that take part in providing vulnerability management functionality. Concepts and relations that existed in the original TRADES Tool (prior to implementing the extension) are denoted using a yellow box and red font. All other elements – including non-red attributes – are part of the new extension. New concepts – *Vulnerability*, *Rule*, and *ComponentType* – extend the original tool's ontology, which featured the vulnerability management related concepts: *Analysis*, *Component* and *Control*. New relations between new concepts and themselves as well as between new concepts and previously-implemented concepts are added, e.g., "componentTypesAffected" between *Rule* and *ComponentType* and "controls" between *Rule* and *Control*. New attributes and operations are added to previously implemented concepts, e.g., "vulnerable" and "mitigated" in *Component*. Some of the new relations and attributes are derived, i.e., their value is automatically computed by the tool based on other model elements. Derived metamodel elements are identified by a slash (/) prefix in their name.

The metamodel's extension provides the core support for vulnerability management in systems security modelling. The new *Vulnerability* concept allows to accommodate various vulnerability definitions within any specific model. These can be either class-level definitions (weaknesses) or implementation-level definitions (vulnerabilities). The *manifests* relation between one *Vulnerability* and another can capture the hierarchical organisation between vulnerability classes, as offered by the CWE knowledge base; as well as the relation between an implementation vulnerability (CVE) and the class that it manifests (CWE).

The new *Rule* concept allows to bind *Vulnerability*, *Component Type* and *Control* elements in support of defining meaningful security rules. This binding is interpreted as the specific set of controls (*Rule.*controls) can be used to mitigate the specific set of vulnerabilities (*Rule.*vulnerabilities*)* in the context of any component that has one of the component types (associated in *Rule.*componentTypesAffected). This is the basis for a fully functional automated reasoning capability. While the full details of the automated reasoning mechanism are beyond the scope of this article, we note that the tool's automated reasoning results are reflected in some of the derived attributes and relations. For example, the Boolean "vulnerable" attribute of a *Component* element indicates if there are any vulnerabilities that are relevant to the component yet are unmitigated, and the "unmitigatedVulnerabilities" relation between *Component* to *Vulnerability* allows indicating the set of vulnerabilities that might affect a specific component and are unmitigated.

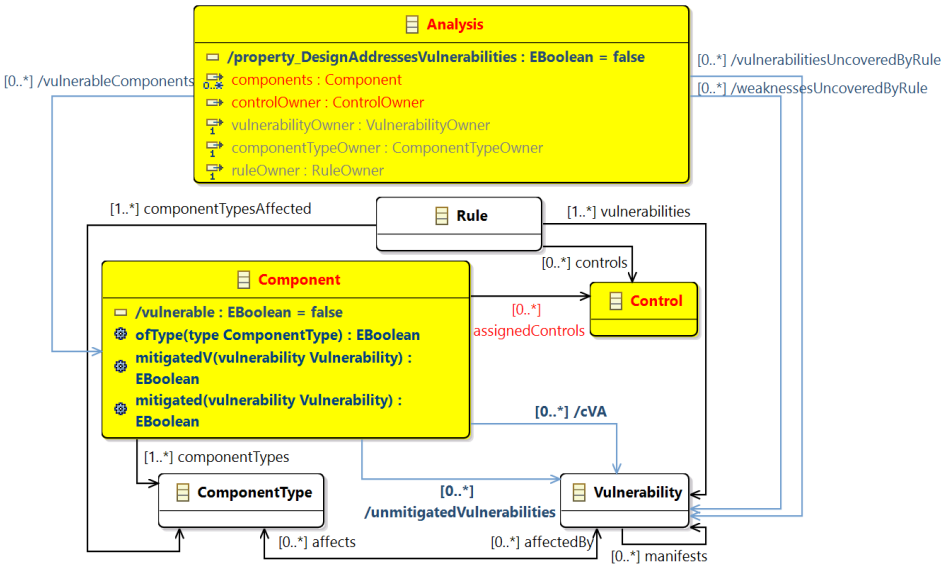

Figure 1: TRADES Tool metamodel extension.

TRADES Tool is an integrative modelling platform [9]. When extending it for vulnerability management purposes, we considered it essential to integrate pertinent information from the expert knowledge bases, namely CWE and CVE. Accordingly, we introduced new tool functionalities to import CWE and CVE information into modelling projects.

Importing the CWE catalogue is approached similarly to the previously implemented Security Controls catalogue import functionality, offering a consistent user experience. Figure 2 shows the main dialogue screen of the CWE import functionality. It allows to select a CWE catalogue (in the form of an XML file) and import it into a selected modelling project. Once the import is completed, the entire CWE catalogue is available for use, with every weakness catalogue entry being a *Vulnerability* element. Figure 3 shows the Model Explorer panel of TRADES Tool, with a CWE catalogue being available.

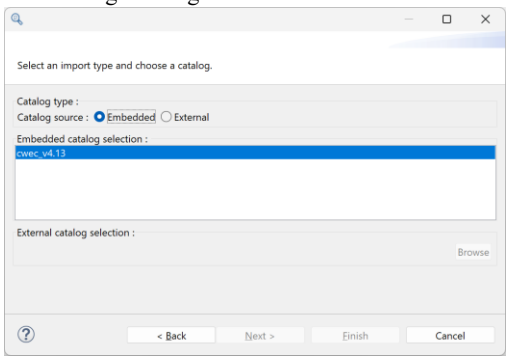

Figure 2: Import CWE catalogue dialogue.

Importing CVE information required a different strategy, due to the frequent disclosure of new vulnerabilities and the volume of CVE records. Instead of importing a complete catalogue of CVE vulnerabilities, which would have overloaded a specific modelling project, we allow a user to import specific vulnerabilities that relate to a specific design. We use the NIST Vulnerabilities API to query the NVD knowledge base and retrieve CVE vulnerabilities. We provide two options for that. Figure 4 shows the first option, in which component types that are defined in the design (model) can be used to import the vulnerabilities that affect them. For this, we use Common Platform Enumeration (CPE) definitions. CPE is a structured naming scheme for information technology systems, software, and packages. A CPE name can be specified as a component type within TRADES models. The NVD API supports the retrieval of CVEs that affect specific CPEs. Figure 5 shows the second option, in which a user can create an API query to retrieve a set of vulnerabilities. In each of the cases, the resulting set of vulnerabilities is shown to the user, who can then decide which of the vulnerabilities to import into the modelling project. Once they are imported, they appear as a catalogue, alongside other catalogues (such as the CWE and the Security Controls catalogue). Figure 6 shows the "Relevant CVEs" catalogue, next to the imported CWE and Security Controls (NIST SP 800-53) catalogues.

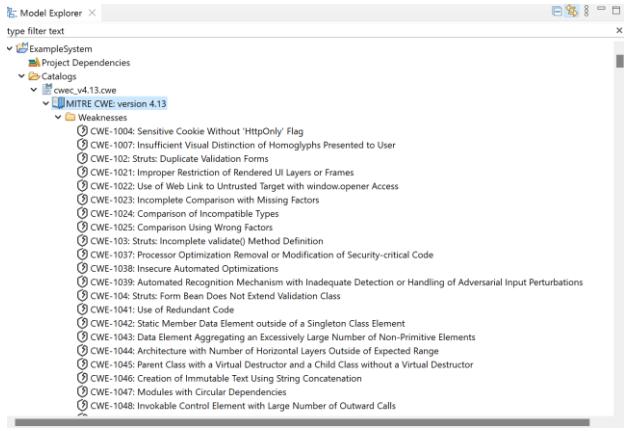

Figure 3: CWE catalogue incorporated into a modelling project.

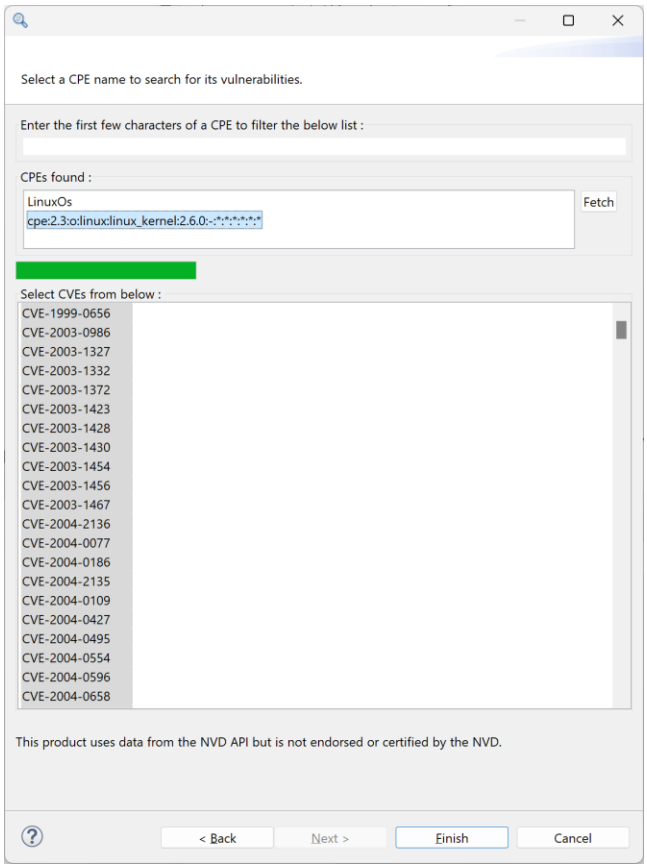

Figure 4: Importing CVE records using component types available in the model.

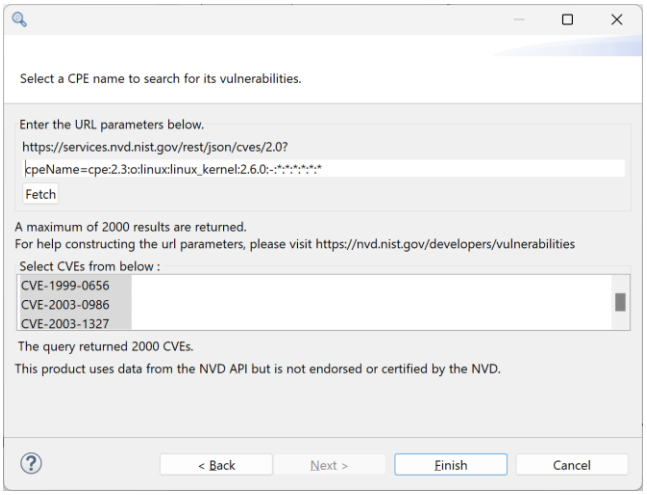

Figure 5: Importing CVE records using custom queries.

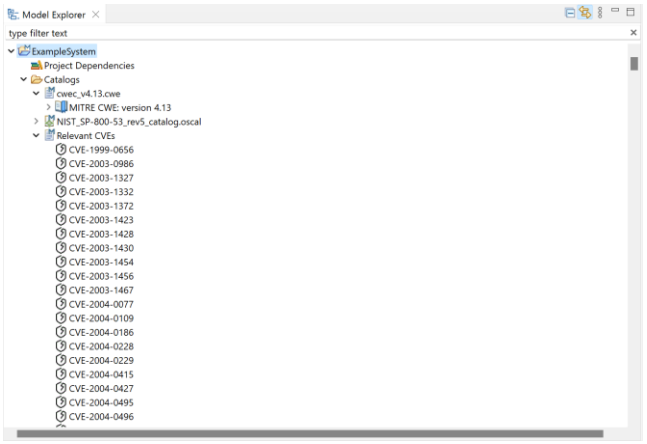

Figure 6: Imported CVE records available as a catalogue within a modelling project.

Figure 7 demonstrates the results of the automated reasoning mechanism. The automated reasoning mechanism is applied continuously, on-the-fly, based on the metamodel-compliant design model, and is therefore fully synchronised with the model of the system security design. The figure shows the attributes of a *Component* element named LinuxInstance. The element has a component type corresponding to a Linux version (based on CPE), and it is assigned a security control named Global Control. The assignment of this control mitigates many vulnerabilities and weaknesses (some shown in CVA and CWA derived attributes; CVA stands for Collection of Vulnerabilities Associated with a component, and CWE stands for Collection of Weaknesses Associated with a component), based on a pre-defined rule (Rule Global). However, some vulnerabilities and weaknesses remain (evident in the Unmitigated Vulnerabilities and Unmitigated Weaknesses attributes), resulting in the component being assessed as vulnerable (*true* value for the Vulnerable derived attribute).

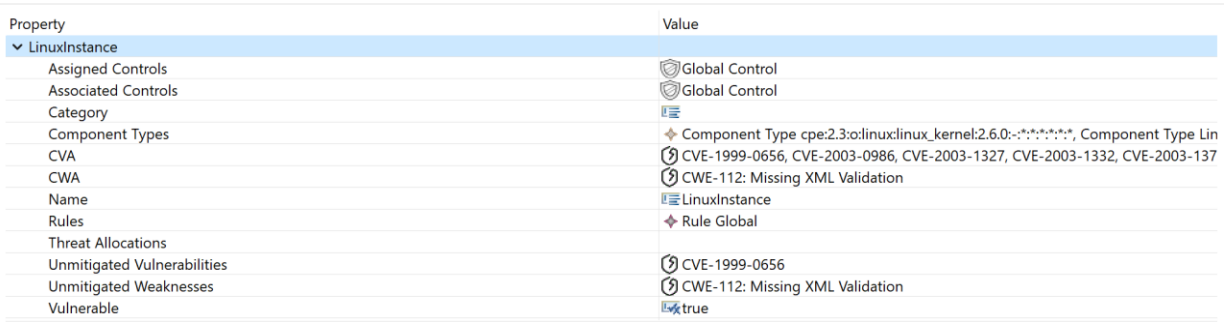

Figure 7: Various attributes of a component within a model, including automated reasoning results.

4 Conclusion

The trustworthiness-critical effort of establishing and maintaining the vulnerability posture of systems is not supported by systems design modelling tools. This hinders the ability to secure systems by design as well as the ability to reason about the security of systems.

We developed a new extension that is seamlessly integrated into an open-source systems security design and assessment modelling tool. This extends the modelling tool with vulnerability management related concepts and features, including vulnerability-related rule specification, importing vulnerability-related information from widely used knowledge bases and employing automated reasoning to reason about the security of systems designs at scale. We intend to integrate a reporting functionality that will allow to report the automated reasoning mechanisms' findings, so that pertinent information can be communicated to various stakeholders, particularly decision-makers such as executive managers.

Our open-source offering can be leveraged by security researchers who wish to introduce and exercise rigorous and scalable modelling artifacts into their research. Both researchers and practitioners can adapt the tool for their specific needs, while relying on the tool as a research-informed infrastructure.

The extended tool allows to integrate information from multiple knowledge bases and use the information in an integrated manner to solve domain problems, e.g., specifying security rules as an organisational policy and reasoning about the vulnerability posture of a system throughout its development and operation. These capabilities enable to use the extended modelling tool for educating security researchers and students about vulnerability management, and specifically how it integrates into security analyses and security by design practices.

ACKNOWLEDGMENTS

The authors wish to thank System C for its role in implementing the extended tool.

This work is funded by Innovate UK, grant number 75243.

REFERENCES

[1] CWE - Common Weakness Enumeration website: *https://cwe.mitre.org/*.

[2] Cybersecurity & Infrastructure Security Agency 2023. *Secure-by-Design Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Secure by Design Software*.

[3] Cybersecurity & Infrastructure Security Agency 2023. *The Case for Memory Safe Roadmaps*.

[4] Huff, P. and Li, Q. 2021. Towards Automated Assessment of Vulnerability Exposures in Security Operations. (2021), 62–81.

[5] Khalil, S.M. et al. 2024. Threat modeling of industrial control systems: A systematic literature review. *Computers & Security*. 136, (Jan. 2024), 103543. DOI:https://doi.org/10.1016/j.cose.2023.103543.

[6] McGraw, G. 2012. Software Security. *Datenschutz und Datensicherheit - DuD*. 36, 9 (Sep. 2012), 662–665. DOI:https://doi.org/10.1007/s11623-012-0222-3.

[7] MITRE CVE website: *https://cve.mitre.org/*.

[8] Shaked, A. 2023. A model-based methodology to support systems security design and assessment. *Journal of Industrial Information Integration*. 33, (Jun. 2023), 100465. DOI:https://doi.org/10.1016/j.jii.2023.100465.

[9] Shaked, A. 2024. Facilitating the Integrative Use of Security Knowledge Bases within a Modelling Environment. *Journal of Cybersecurity and Privacy*. 4, 2 (Apr. 2024), 264–277. DOI:https://doi.org/10.3390/jcp4020013.

[10] Syed, R. 2020. Cybersecurity vulnerability management: A conceptual ontology and cyber intelligence alert system. *Information & Management*. 57, 6 (Sep. 2020), 103334. DOI:https://doi.org/10.1016/j.im.2020.103334.

[11] Varela-Vaca, Á.J. et al. 2023. Feature models to boost the vulnerability management process. *Journal of Systems and Software*. 195, (Jan. 2023), 111541. DOI:https://doi.org/10.1016/j.jss.2022.111541.

[12] Vulnerabilities API at NVD: *https://nvd.nist.gov/developers/vulnerabilities*. Accessed: 2024-07-08.

[13] Vulnerability management Guidance: *https://www.ncsc.gov.uk/collection/vulnerability-management*. Accessed: 2024-07-08.

[14] Walkowski, M. et al. 2021. Vulnerability Management Models Using a Common Vulnerability Scoring System. *Applied Sciences*. 11, 18 (Sep. 2021), 8735. DOI:https://doi.org/10.3390/app11188735.