A Multi-Dimensional Model and Algorithm for the Classification and Remediation of Software Vulnerabilities

MASTER THESIS

Felix Berger

Submitted on 3 April 2025

Friedrich-Alexander-Universität Erlangen-Nürnberg
Faculty of Engineering, Department Computer Science
Professorship for Open Source Software

Supervisor:
Martin Wagner, M. Sc.
Prof. Dr. Dirk Riehle, M.B.A.

Friedrich-Alexander-Universität
Faculty of Engineering

Declaration of Originality

Erlangen, 3 April 2025

License

Erlangen, 3 April 2025

Abstract

This master's thesis addresses shortcomings in the vulnerability management component of the Software Composition Analysis (SCA) Tool developed by the Professorship for Open Source Software (OSS) at the Friedrich-Alexander University Erlangen-Nürnberg (FAU), by proposing a multidimensional classification and remediation framework for software vulnerabilities. The developed approach integrates the Common Vulnerability Scoring System (CVSS) for technical impact assessment with the Exploit Prediction Scoring System (EPSS) for real-world exploit likelihood, offering a balanced view of both intrinsic risk and active threats. A model and algorithm are introduced to compute contextual classification scores, complemented by a stakeholder-specific remediation strategy leveraging the Stakeholder-Specific Vulnerability Categorization (SSVC) framework. Additionally, a rank-ordering model prioritizes vulnerabilities, ensuring critical and data-incomplete vulnerabilities are given immediate attention. The implementation utilizes data sources such as Open Source Vulnerabilities (OSV) and incorporates robust caching and daily refresh mechanisms to minimize unnecessary traffic and enhance performance. The effectiveness and practical applicability of the framework are confirmed through evaluations conducted by domain experts.

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
CAPEC	Common Attack Pattern Enumeration and Classification
CISA	U.S. Cybersecurity and Infrastructure Security Agency
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DST	Debian Security Tracker
EPSS	Exploit Prediction Scoring System
EU	European Union
FAU	Friedrich-Alexander University Erlangen-Nürnberg
FIRST	Forum of Incident Response and Security Teams'
GAD	GitHub Advisory Database
GDPR	General Data Protection Regulation
GHSA	GitHub Security Advisory
MDE	Microsoft Defender for Endpoint
MITRE	MITRE Corporation
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OSS	Professorship for Open Source Software
OpenSSF	Open Source Vulnerability Format
OSV	Open Source Vulnerabilities

POC	Proof-Of-Concept
R7IVM	Rapid7 InsightVM
RHD	Red Hat CVE Database
RRS	Real Risk Score
SCA	Software Composition Analysis
SBOM	Software Bill of Materials
SSVC	Stakeholder-Specific Vulnerability Categorization
TVM	Tenable Vulnerability Management
VA	VulnAware
VMDR	Qualys Vulnerability Management, Detection, and Response
VPR	Vulnerability Priority Rating

1 Introduction

In today's software development landscape, the use of open-source components is ubiquitous. According to the Open Source Security and Risk Analysis Report 2024 by Black Duck Software, Inc (2024), 96% of codebases analyzed contain open-source components. While this practice accelerates development processes and fosters innovation, it also introduces significant security risks. According to the 2023 State of Open Source Security Report by Snyk Limited (2023), 87% of organizations were impacted by one or more supply chain security issues in the past year. Specifically, 53% had to patch one or more vulnerabilities, and 61% implemented new tooling and practices to better handle supply chain vulnerabilities. This highlights how frequently vulnerabilities in open-source software are exploited and the significant risks they pose.

The complexity of modern software projects leads to extensive dependency graphs that are difficult to oversee. A single project can utilize hundreds of open-source libraries, each bringing its own dependencies. This complexity makes it challenging to track and manage vulnerabilities across the entire software supply chain.

A notable example highlighting the limitations of single-dimensional vulnerability scoring is the Heartbleed bug in the OpenSSL library.[1] At the time of its discovery in 2014, vulnerabilities were primarily assessed using CVSS version 2 (Balbix, Inc., 2020). Heartbleed received a relatively moderate CVSS v2 base score of 5.0 (Medium) on a scale of 0 to 10, yet was rapidly and widely exploited, leading to the leakage of sensitive information (e.g., private keys, passwords) from millions of servers worldwide.[2] This discrepancy between the moderate numerical rating and its significant real-world impact clearly illustrates why incorporating additional dimensions, such as real-world exploitability, is crucial for accurate prioritization in vulnerability management frameworks.

SCA tools and Software Bill of Materials (SBOM)s have established themselves as instruments for providing transparency about the components used and their security status. While SBOMs allow developers to maintain a comprehensive

[1]https://heartbleed.com/

[2]https://levelblue.com/blogs/security-essentials/cvss-score-a-heartbleed-by-any-other-name

inventory of all software components, SCA tools actively check these components for known vulnerabilities. However, existing solutions often reach their limits when it comes to classifying newly discovered vulnerabilities and assessing their impact on a specific software project.

The OSS at the FAU develops its own SCA Tool,[3] which aims to facilitate the secure, efficient, and regulatory-compliant use of open-source software within modern software engineering projects. Specifically, the tool addresses three critical domains:

- **Governance:** Assurance that only approved open-source licenses are utilized, thereby aiding organizations in adhering to internal policies and mitigating potential legal risks.
- **Compliance:** Simplification of the generation of legal notices for license-compliant distribution of software products, reducing the complexity and overhead of adhering to license requirements.
- **Vulnerability Management:** Provision of continuous monitoring of open-source code for newly discovered vulnerabilities, delivering actionable intelligence to mitigate risks associated with software dependencies.

VulnAware (VA) serves as the precursor of the vulnerability management component of the SCA Tool developed by the OSS at the FAU. It addresses the challenges of maintaining transparency and control over vulnerabilities in complex software dependency graphs by accepting SBOM files and continuously checking the contained components for known vulnerabilities. Through a web interface, developers are presented with the components they use, potential risks, and possible remediation measures. However, VA currently lacks an mechanism to generate tailored remediation recommendations based on the specific software context. The scoring system uses the CVSS to sort vulnerabilities by urgency (Nehrke, 2023). Despite its structured approach, CVSS faces criticism for assigning numerical values to qualitative data without sufficient empirical justification, leading to inconsistent and sometimes misleading scores. Studies have shown high variability in CVSS scoring among professionals, with discrepancies of 2–4 points on a scale from 0 to 10 being common (J. Spring et al., 2021).

To overcome these limitations, this master's thesis extends the vulnerability management component of the SCA Tool by developing an extended model for the multidimensional classification of vulnerabilities and providing tailored remediation recommendations. This includes:

- A model for multidimensional classification of vulnerabilities.
- An algorithm to compute a classification for a known vulnerability in the

[3]https://scatool.com/about/

context of a given software.

- An algorithm to make recommendations about how to remedy them.
- A model to rank-order all known classified vulnerabilities.

By extending the vulnerability management component of the SCA Tool, this thesis aims to improve vulnerability classification by addressing the limitations of CVSS's numerical scoring, incorporating multidimensional assessment criteria, and providing actionable remediation. These enhancements will improve usability and prioritization accuracy.

2 Literature review

To enhance the functionality of vulnerability management in SCA Tool, it is essential to conduct a thorough review of existing research in vulnerability classification and remediation. This chapter provides an overview of current models, tools, and approaches in the field. The insights gained serve as the foundation for the development of a multidimensional classification and remediation model, as well as algorithms to assess, prioritize, and recommend remediation actions for vulnerabilities in software projects. The findings from this literature review directly inform the multidimensional model developed in chapter 4.

2.1 Existing Vulnerability Identification and Classification Models

Vulnerability classification models provide structured methods for assessing and prioritizing security vulnerabilities based on factors such as severity, exploitability, and potential impact.

Given that the vulnerability management system in SCA Tool retrieves vulnerability information from the OSV Database, which heavily relies on the Common Vulnerabilities and Exposures (CVE) system for uniquely identifying vulnerabilities, it is essential to include CVE in this discussion. CVE acts as the foundational identification system that standardizes the naming of vulnerabilities, enabling consistent referencing across databases, tools, and classification models. Although CVE itself does not classify vulnerabilities, it provides the standardized identifiers necessary for databases and tools to organize and retrieve vulnerability information consistently (MITRE Corporation, 2024).

Models such as CVSS and EPSS offer standardized approaches to vulnerability classification. Classifications generated by these models are typically stored in vulnerability databases, linked by their respective CVE identifiers. Tools and systems implementing these classification models subsequently access these databases to evaluate vulnerability severity and exploitability.

CVSS has been selected for its structured scoring system that assesses vulnerabilities according to their severity. However, recognizing the limitations of CVSS (see section 2.1.2), EPSS is incorporated to enrich prioritization through dynamic threat intelligence. Additionally, SSVC is integrated due to its consideration of stakeholder-specific factors. Using tailored decision trees, SSVC guides context-sensitive prioritization and remediation decisions, helping organizations choose appropriate responses such as patching, monitoring, or deprioritizing vulnerabilities, based on their operational contexts and resources.

Other models, such as Common Weakness Enumeration (CWE)[1] and Common Attack Pattern Enumeration and Classification (CAPEC)[2], primarily focus on categorizing software weaknesses and attack patterns rather than directly classifying vulnerabilities. While these models are valuable in broader security contexts, they are less directly applicable to the specific goals of vulnerability classification and remediation within SCA tools.

In summary, the vulnerability management system in SCA Tool builds upon the foundational identification provided by CVE, using it to retrieve comprehensive vulnerability details from relevant databases, which are then leveraged by classification models to support effective vulnerability assessment and remediation. The following sections present these classification models and their underlying concepts in greater detail.

2.1.1 Common Vulnerabilities and Exposures

The CVE system provides a well-established and standardized framework for uniquely identifying and referencing publicly disclosed cybersecurity vulnerabilities. It is maintained by the MITRE Corporation (MITRE) and funded by the U.S. Cybersecurity and Infrastructure Security Agency (CISA). Each vulnerability in the CVE system is assigned a unique identifier, such as `CVE-2024-12345`, which acts as a universal reference point for tools, databases, and discussions related to cybersecurity. The CVE entries contain basic metadata, such as the vulnerability description, affected products, and references to further details. However, CVE itself does not include technical details, exploit code, or remediation instructions. Instead, it serves as a reference system that links to external sources for further information.

The structure of a CVE identifier follows the format `CVE-YYYY-NNNNN`, where:

- **YYYY (e.g., 2024)**: Indicates the year in which the CVE-ID was assigned or reserved. This helps provide a temporal context for when the vulnerability became publicly known or cataloged.

[1]https://cwe.mitre.org/
[2]https://capec.mitre.org/

- **NNNNN (e.g., 12345)**: Represents a unique, sequential number that identifies the vulnerability within the specified year. This number is assigned by the CVE system to ensure uniqueness.

For example, in the identifier `CVE-2024-12345`:

- `2024` indicates that the CVE-ID was assigned or cataloged in the year 2024.
- `12345` is the specific, unique number that distinguishes this vulnerability from others cataloged in the same year.

The primary objective of CVE is to improve the coordination and sharing of vulnerability information across different organizations, tools, and platforms. By providing a consistent and unique identifier for each vulnerability, CVE enables organizations to align their vulnerability management processes, ensuring that the same vulnerability is referenced accurately in security tools, advisories, and incident response processes.

While CVE is not a classification or scoring system, it enables security tools and vulnerability management systems to retrieve vulnerability data, such as CVSS and EPSS, from databases by providing a standardized identification mechanism (MITRE Corporation, 2024).
In the implementation (see section 6.1.1), the CVE system acts as the primary identifier used to retrieve corresponding vulnerability details from external databases such as National Vulnerability Database (NVD) and `first.org`'s EPSS, facilitating standardized data integration into the vulnerability management workflow.

2.1.2 Common Vulnerability Scoring System

The CVSS provides a method to capture the essential characteristics of a vulnerability, reflecting its severity to help organizations assess and prioritize their vulnerability management processes. However, the scoring algorithm lacks formal and empirical justification, and its creators caution against using CVSS as a risk score, though some compliance bodies, such as the U.S. government and the global payment card industry, explicitly recommend this misuse (J. Spring et al., 2021).

The CVSS scoring framework (Version 3.1, currently the most widely adopted version)[3] is structured into three metric groups: Base, Temporal, and Environmental. Each group contributes uniquely to the final score (FIRST, 2025):

- **Base Metrics**: Capture characteristics of a vulnerability that remain constant over time and across environments, such as access complexity, au-

[3]https://vulncheck.com/blog/common-vulnerability-scoring-system

thentication requirements, and impacts on confidentiality, integrity, and availability.

- **Temporal Metrics**: Reflect aspects that can change over time, like the availability of exploit code or patches, which adjust the base score to represent the current level of exploitability.
- **Environmental Metrics**: Allow organizations to adjust the score based on their specific context, incorporating factors like potential collateral damage and the prevalence of affected systems, providing a customized risk assessment more reflective of the organization's environment.

Despite its structured approach, the CVSS scoring system faces criticism for several significant limitations. The process of converting qualitative evaluations into a numerical score between 0 and 10 has been critiqued for assigning numerical values to ordinal data without sufficient empirical justification. This methodology can produce inconsistent and sometimes misleading scores, as it overlooks important contextual factors. Additionally, studies have shown high variability in CVSS scoring even among experienced security professionals, with score discrepancies of 2–4 points being common. Such discrepancies are substantial, given that four points span the entire "high" severity range, highlighting a lack of precision in the system (J. Spring et al., 2021).

Evolution of CVSS Versions

According to Balbix, Inc. (2020), the CVSS has evolved over time to address the limitations of earlier versions. Each new version introduces improvements to enhance the accuracy and usability of vulnerability assessments.

CVSS Version 1 The first version of CVSS was the initial attempt to standardize vulnerability scoring but lacked the necessary granularity and flexibility. It did not adequately represent factors like attack complexity or impacts on integrity and availability, leading to limited adoption.

CVSS Version 2 The second version improved upon the first version by introducing more detailed metrics and a better framework for assessing vulnerabilities. It considered aspects like access vectors, access complexity, and authentication requirements. However, it still faced challenges in accurately representing modern vulnerabilities, particularly regarding the complexity of required privileges and user interactions.

CVSS Version 3 The third generation further refined the scoring system by adding new metrics and modifying existing ones for a more precise and context-aware evaluation. It offers detailed considerations of the attack vector and includes metrics for privileges required and user interaction. These improvements help organizations better prioritize risks and develop effective security strategies.

CVSS Version 3.1 CVSS 3.1 builds on the enhancements introduced in Version 3 by refining certain metrics and clarifying guidelines to achieve more accurate and consistent vulnerability assessments. It also standardizes terminology and scoring interpretations, which helps ensure comparability across various systems and tools (FIRST, 2024a). Additionally, as confirmed by current security sources, Version 3.1 remains the officially recognized standard today, enabling organizations to effectively prioritize and address risks.[4]

CVSS Version 4 The latest version, CVSS Version 4.0, has been proposed to address ongoing criticisms but, at the time of writing, has not yet been widely adopted or fully standardized. The most recent revision of its documentation (V1.2, released June 18, 2024) reflects ongoing adjustments and clarifications. Organizations often hesitate to transition to a new major version until it gains broad industry acceptance. Consequently, CVSS Version 3.1 remains the dominant standard for vulnerability assessment (FIRST, 2024b).

CVSS helps assess technical severity, but it does not fully account for security risk, leaving room for improvement. Particularly notable is the documented scoring inconsistency of 2–4 points observed among experienced security practitioners (J. Spring et al., 2021).

This variability can lead to misaligned remediation efforts, emphasizing the need for complementary metrics. To address these limitations, the multidimensional classification model proposed in this thesis (see section 4.1) integrates exploitability indicators such as EPSS, thus achieving a more consistent and risk-oriented prioritization.

2.1.3 Exploit Prediction Scoring System

The EPSS is an open, data-driven framework for assessing the threat posed by software vulnerabilities. It aims to quantify the probability that a vulnerability will be exploited in the wild within the first 12 months after its public disclosure. Unlike traditional methods that often rely on subjective expert opinions or severity scores like the CVSS, EPSS utilizes objective, publicly available data and machine learning to make accurate predictions.

EPSS employs a logistic regression model that is simple to implement and interpret. By reducing the number of required input variables and focusing on publicly accessible data sources, it offers high predictive accuracy regarding the likelihood of exploitation. Key factors in the model include:

[4]https://vulncheck.com/blog/common-vulnerability-scoring-system

- **Availability of Exploit Code**: The presence of publicly available exploit code or Proof-Of-Concept (POC) exploits increases the likelihood of real-world exploitation.
- **Affected Software Vendors**: Vulnerabilities in software from widely used vendors, such as Microsoft or Adobe, have a higher probability of being exploited due to their prevalence.
- **Vulnerability Age**: The age of the vulnerability since its public disclosure, as older vulnerabilities can show differing patterns of exploitation over time.
- **Attack Complexity and User Interaction**: Although not directly inputted into EPSS, historical data reflects the impact of these factors, with simpler attack complexities and minimal required user interaction correlating with higher exploit probabilities.
- **Popularity of Affected Software**: Vulnerabilities in widely deployed software are prioritized, as they tend to have a greater impact and higher likelihood of exploitation.
- **Empirical Threat Intelligence**: Data from observed exploitations in real-world scenarios provide a foundation for the EPSS model, enhancing its accuracy by integrating active threat trends.

The system addresses the challenges of prioritization in vulnerability management by enabling security professionals to allocate resources more efficiently and focus on vulnerabilities that are more likely to be exploited.

Overall, EPSS significantly advances security risk assessment by offering an open, data-driven method that directly predicts the likelihood of a vulnerability being exploited. Unlike CVSS, which rates severity based solely on inherent technical characteristics, EPSS employs empirical threat intelligence and a logistic regression model with elastic net regularization to predict real-world exploitability, enabling more effective risk prioritization (Jacobs et al., 2021).

This thesis leverages EPSS to complement CVSS scores within the multidimensional classification model (see section 4.1). Specifically, the proposed algorithm integrates both metrics to generate a composite severity score, enhancing prioritization accuracy by balancing technical severity with realistic exploitation probability (see section 4.2).

2.1.4 Stakeholder-Specific Vulnerability Categorization

The Stakeholder-Specific Vulnerability Categorization (SSVC) is a decision-making framework designed to enhance vulnerability management by focusing on the specific needs and contexts of different stakeholders. Unlike CVSS, which provides a general severity score based on technical characteristics, SSVC uses tailored decision trees to guide organizations through prioritization actions relevant to their unique situations.

SSVC emphasizes the importance of context and the specific roles of organizations in vulnerability management. It recognizes that different stakeholders - such as software vendors, deployers, and coordinators - have diverse priorities and resources. By framing decisions directly through qualitative criteria rather than relying on numerical severity scores, SSVC aims to provide clear and actionable guidance. The decision trees in SSVC lead users through a series of considerations, such as exploitability, exposure, and mission impact, resulting in specific recommended actions.

By incorporating stakeholder-specific factors, SSVC enables organizations to make informed decisions that are better aligned with their operational realities.

Overall, SSVC enhances decision-making by providing stakeholders with a practical framework for determining appropriate remediation actions based on qualitative criteria and their specific organizational contexts, thus improving vulnerability management effectiveness (J. M. Spring et al., 2021).

Within this thesis, the SSVC approach serves as the foundation for the remediation recommendation model (see section 4.3). Specifically, tailored SSVC-based decision trees are implemented to generate clear, stakeholder-specific remediation guidance - such as immediate patching or regular monitoring - aligned explicitly with the needs of developers and security coordinators.

2.2 Existing Multidimensional Approaches and Tools for Vulnerability Detection, Assessment, and Remediation

This section presents well-known tools for vulnerability detection, assessment, and remediation, analyzing their use of multidimensional approaches. These tools are evaluated based on how they integrate factors like severity, exploitability, risk, and remediation strategies to effectively prioritize and address vulnerabilities. Insights into these multidimensional approaches inform the design choices and development of the proposed framework in chapter 4, particularly regarding the integration of real-world exploitability with technical severity and context-sensitive remediation recommendations.

2.2.1 Tenable Vulnerability Management

Tenable Vulnerability Management (TVM) enhances traditional vulnerability assessment by integrating CVSS with its proprietary Vulnerability Priority Rating (VPR). While CVSS provides a standardized method for evaluating technical severity, it lacks consideration of contextual factors and real-world exploitability (J. Spring et al., 2021). Tenable's VPR addresses these limitations by dynamically adjusting scores based on exploit availability, threat intelligence and asset criticality. The VPR provides a single score on a scale from 0.1 to 10.0, with higher values representing a higher likelihood of exploit.

Leveraging its assessments from both CVSS and VPR, TVM offers actionable remediation recommendations tailored to an organization's specific risk profile. Remediation strategies are drawn from different sources like the GitHub Advisory Database (GAD)[5] and the NVD[6]. By incorporating key drivers such as vulnerability age, exploit code maturity, threat intelligence, and technical impact (based on CVSS), TVM ensures vulnerabilities are prioritized based on both technical severity and real-world exploitability. This multidimensional approach allows organizations to address the most critical issues first, ensuring a focus on vulnerabilities that pose the greatest immediate threat (Tenable, Inc., 2024a).

By utilizing proprietary insights and contextual data, the platform guides security teams to mitigate vulnerabilities that are most likely to be exploited, enhancing overall security posture (Tenable, Inc., 2024b).

Inspired by Tenable's integration of contextual factors such as asset criticality and exploit maturity into vulnerability assessments, the multidimensional classification framework developed in this thesis combines technical severity (CVSS) with empirical exploitability (EPSS), resulting in a more context-aware prioritization approach (see section 4.1).

2.2.2 Rapid7 InsightVM

Like TVM, Rapid7 InsightVM (R7IVM) addresses the limitations of CVSS with its proprietary solution, the Real Risk Score (RRS). The RRS augments CVSS by introducing a multidimensional approach that incorporates real-time data from various sources. These dimensions include among other things the CVSS, exploitability, exposure, active threat intelligence, and business impact, which together provide a more dynamic and comprehensive vulnerability assessment. The RRS consolidates these factors into a single score ranging from 1 to 1000, with higher scores indicating higher risk.

The dimension of exploitability is supported by data from the Heisenberg honeypot framework[7], which simulates vulnerable systems and captures attack methodologies. This framework allows the RRS to include real-time exploit data,

[5]https://github.com/advisories
[6]https://nvd.nist.gov/
[7]https://information.rapid7.com/project-heisenberg-cloud.html?CS=blog

focusing on vulnerabilities actively targeted by attackers. In addition, incident reports from Rapid7's Managed Detection and Response team provide confirmed exploitation activity, ensuring that vulnerabilities being targeted in live attacks are prioritized.

The exposure dimension evaluates how accessible a vulnerability is, particularly in terms of public exposure. The RRS considers whether assets are internet-facing or otherwise accessible, as this increases the potential risk of exploitation. This assessment is enhanced by intelligence partnerships, providing insights into vulnerabilities actively exploited across various sectors.

Another key dimension is business impact, which is determined through a tagging system within R7IVM. This allows organizations to adjust the prioritization of vulnerabilities based on the criticality of the affected assets, such as systems that host sensitive data (Rapid7, 2017).

Similar to Tenable (see section 2.2.1), R7IVM integrates technical severity with factors like real-world exploitability and business impact, enhancing vulnerability prioritization through a multidimensional approach. This aligns with the multi-dimensional model proposed in this thesis, combining CVSS and EPSS to achieve context-aware prioritization (see section 4.1).

2.2.3 Snyk

Snyk is a platform for vulnerability detection and remediation that employs a multidimensional approach. In addition to traditional vulnerability detection, Snyk monitors multiple vulnerability databases, such as CVEs from the NVD and others, to ensure comprehensive coverage of known vulnerabilities. Snyk also tracks user activity on GitHub, including issues, pull requests, and commit messages that may indicate the presence of a security flaw. This is complemented by tools that identify recurring security patterns across open-source packages, as well as manual audits conducted by the Snyk Security team to scrutinize widely used packages for vulnerabilities. To facilitate prioritization, Snyk provides two types of scores: a Priority Score (1-1000) for all Snyk products, and a Risk Score (1-1000) for Snyk Open Source[8] and Snyk Container[9] (Snyk Limited, 2024c, 2024d).

Furthermore, Snyk maintains a proprietary vulnerability database that extends publicly available data by including vulnerabilities that may not yet be publicly disclosed, thus allowing for early detection of security issues. This combination of multiple data sources enhances the breadth and depth of Snyk's detection capabilities (Snyk Limited, 2024b).

[8]https://docs.snyk.io/scan-with-snyk/snyk-open-source
[9]https://docs.snyk.io/scan-with-snyk/snyk-container

In its vulnerability assessment, Snyk does not rely solely on CVSS scores. It incorporates additional dimensions such as the availability of fixes, exploit maturity (whether an exploit is available and how advanced it is), and the popularity of the affected components. This multidimensional assessment allows for more accurate prioritization of vulnerabilities based on their real-world impact (Snyk Limited, 2024a).

In addition to data from the NVD, Snyk utilizes its own proprietary vulnerability database to deliver a comprehensive, multidimensional approach for vulnerability detection, assessment, and remediation. Similar to Tenable (see section 2.2.1) and Rapid7 (see section 2.2.2), Snyk incorporates additional factors such as exploit maturity, availability of fixes, and the popularity of affected components, enabling more effective vulnerability prioritization (Snyk Limited, 2024a).

This thesis follows a similar multidimensional approach by integrating standardized technical severity scores (CVSS) with empirical exploit probabilities (EPSS) to enhance vulnerability prioritization (see section 4.1). Furthermore, akin to Snyk's consideration of exploit maturity and patch availability, the proposed remediation recommendation algorithm (see section 4.4) includes context-specific factors such as asset criticality and patch availability, employing tailored SSVC-based decision trees for generating actionable recommendations.

2.2.4 Other Multidimensional Approaches

In addition to the tools discussed, other platforms have adopted multidimensional approaches to vulnerability assessment, incorporating similar metrics and dimensions. For example, Microsoft Defender for Endpoint (MDE) and Qualys Vulnerability Management, Detection, and Response (VMDR) each utilize multidimensional frameworks that address factors like exploitability, threat intelligence, and asset criticality. These platforms, while varying in methodology, fundamentally rely on the same core principles - prioritizing vulnerabilities based on contextual risk factors and real-world impact. Consequently, the multidimensional approaches presented here provide a representative overview of current practices in comprehensive vulnerability assessment (Microsoft, 2024; Qualys, 2022).

2.3 Vulnerability Databases and Remediation Resources

This section introduces well-known vulnerability databases that play a crucial role in identifying, classifying, and guiding the remediation of software vulnerabilities. These databases provide standardized information and recommended actions that help prioritize and address vulnerabilities across various software ecosystems. The databases described here serve as key data sources for the multidimensional classification model presented in chapter 4, particularly for obtaining CVSS and EPSS scores necessary for calculating comprehensive severity scores and providing informed remediation recommendations.

2.3.1 Open Source Vulnerabilities

The OSV project provides a platform for identifying known vulnerabilities in third-party open-source dependencies, enabling developers to prioritize remediation efforts based on the impact of these vulnerabilities. The OSV infrastructure aggregates data from multiple vulnerability databases that adhere to the Open Source Vulnerability Format (OpenSSF)[10], ensuring consistency and interoperability. By using bisection and version analysis, OSV represents affected versions for each vulnerability, which aids in applying targeted remediation measures. The project aggregates data from sources including the GAD, PyPI[11], and the Go Vulnerability Database[12] (OSV, 2024).

Within this thesis, the OSV database serves as the primary source for retrieving vulnerability information, enabling efficient access to standardized vulnerability data and supporting the multidimensional classification and remediation models implemented in section 6.1.1.

2.3.2 National Vulnerability Database

The NVD, operated by the National Institute of Standards and Technology (NIST), is a comprehensive repository of known vulnerabilities, primarily utilizing data from the CVE program. After each CVE entry is published, typically within an hour, the NVD enriches it by assigning CVSS scores, which indicate the ease of exploitation and potential impact, and categorizing vulnerabilities with CWE identifiers. This enrichment helps users assess the severity, scope, and potential impact of vulnerabilities on affected software and hardware configurations.

While the NVD itself does not provide direct remediation recommendations, it enhances CVE data with reference tags and links to additional resources that may include vendor advisories or relevant patches, allowing organizations to determine potential mitigation steps. Through consistent updates, quality assurance, and community feedback, the NVD provides an accessible and regularly updated dataset that supports security professionals in identifying, classifying, and managing vulnerabilities across a wide range of platforms (NIST, 2024).

[10]https://ossf.github.io/osv-schema/
[11]https://github.com/pypa/advisory-database
[12]https://github.com/golang/vulndb

In this thesis, the NVD is utilized as a key external data source for obtaining detailed CVSS vectors whenever local vulnerability data in the OSV database is incomplete, directly supporting the multidimensional vulnerability scoring mechanism implemented in section 6.1.1.

2.3.3 GitHub Advisory Database

The GAD[13] is an extensive resource for tracking security vulnerabilities in open-source projects, compiled from multiple sources, including advisories - official notifications about security vulnerabilities or malicious software, providing details on affected packages, versions, and potential mitigation measures - reported directly on GitHub and contributions from external databases such as the NVD, npm Security Advisories[14], and language-specific sources like RustSec[15]. Each advisory undergoes classification and may be grouped as GitHub-reviewed, unreviewed, or malware-related, thus providing users with clear distinctions regarding the trustworthiness and origin of the data.

Each advisory in the GAD is uniquely identified by a GitHub Security Advisory (GHSA), such as `GHSA-xxxx-yyyy-zzzz`. The GHSA ensures that advisories are uniquely referenced within the GitHub ecosystem, providing a consistent mechanism for identifying and addressing vulnerabilities reported by maintainers or the GitHub community.[16]

Advisories within the database are standardized using the OSV format, which includes key details such as the affected ecosystem, package, impacted versions, severity, and in some cases, patched versions. GitHub enhances vulnerability entries with CVSS metrics for severity and impact. For advisories with applicable data, the database also includes EPSS scores, providing a probabilistic measure of exploit likelihood, which aids organizations in prioritizing vulnerability response efforts.

In addition to verified security advisories, GitHub includes unreviewed advisories imported directly from the NVD, allowing users to access a broader scope of known vulnerabilities. Notably, malware advisories, exclusive to the npm ecosystem, inform users about intentionally malicious code packages, typically linked to substitution attacks. While GitHub doesn't directly verify these malware reports, their inclusion in the advisory database helps security teams identify and remove such threats.

By integrating advisories from a range of ecosystems - including popular package registries - the GitHub Advisory Database serves as a centralized, reliable source

[13]https://github.com/github/advisory-database
[14]https://www.npmjs.com/search?q=advisories
[15]https://rustsec.org/
[16]https://github.com/github/advisory-database

for open-source vulnerability management (GitHub, Inc., 2024).

Within this thesis, the GAD is part of the aggregated data provided by the local OSV database, used primarily to retrieve CVE identifiers and corresponding CVSS vectors, essential for vulnerability classification and scoring (see section 6.1.1).

2.3.4 FIRST.org EPSS Database and API

The Forum of Incident Response and Security Teams' (FIRST) EPSS database provides a comprehensive platform for accessing EPSS scores, which estimate the likelihood of a vulnerability being exploited in real-world scenarios. This resource is accessible via an Application Programming Interface (API), allowing users to programmatically query and integrate EPSS scores into their vulnerability management workflows.

The API supports queries by CVE identifiers, enabling organizations to retrieve specific scores for vulnerabilities of interest. Additionally, the API provides metadata such as the date of the score calculation and supplementary documentation to guide implementation. The database is regularly updated to reflect the latest data and predictive models, ensuring accurate and actionable insights FIRST, 2024c.

Unlike the theoretical foundation of EPSS discussed earlier (see section 2.1.3), this section focuses on the practical implementation and the type of data collected to support EPSS predictions. The following data points are integrated into the EPSS database:

- **Vendor Information**: Extracted from the CPE (Common Platform Enumeration)[17] via the NVD.
- **Vulnerability Age**: Measured in days since the CVE was published in the MITRE CVE list[18].
- **References with Categorical Labels**: Links to sources such as the MITRE CVE List and NVD, which categorize the content of each reference.
- **Normalized Multiword Expressions**: Extracted from the vulnerability descriptions in the MITRE CVE List to enhance semantic analysis.
- **Weakness Information**: Collected from CWE identifiers via the NVD.
- **CVSS Metrics**: Includes the base vector from CVSS 3.x, obtained via the NVD.

[17]https://nvd.nist.gov/products/cpe
[18]https://cve.mitre.org/cve/search_cve_list.html

- **CVE Presence on Public Lists**: Information about whether a CVE is listed on notable platforms, such as CISA KEV[19], Google Project Zero [20], or Trend Micro's Zero Day Initiative (ZDI)[21], among others.
- **Publicly Available Exploit Code**: Sourced from platforms like Exploit-DB[22], GitHub[23], and MetaSploit[24].
- **Offensive Security Tools and Scanners**: Integration with tools such as Intrigue[25] and sn1per,[26] to identify exploit capabilities.

Another aspect of EPSS is timing information; for example, tracking when a Metasploit module for a CVE was added to correlate it with real-world exploitation activity (FIRST, 2024c).

To better understand the distribution of EPSS probabilities across known CVEs, figure 2.1 shows a histogram based on the EPSS data as of 2022-03-04. The majority of CVEs have a very low exploitation probability, with most values concentrated near 0%. This indicates that while a large number of vulnerabilities exist, only a small fraction are likely to be exploited in real-world scenarios.

However, as shown in the tail of the distribution, a subset of vulnerabilities has a significantly higher likelihood of exploitation. These higher-probability vulnerabilities represent critical risks that organizations should prioritize for remediation.

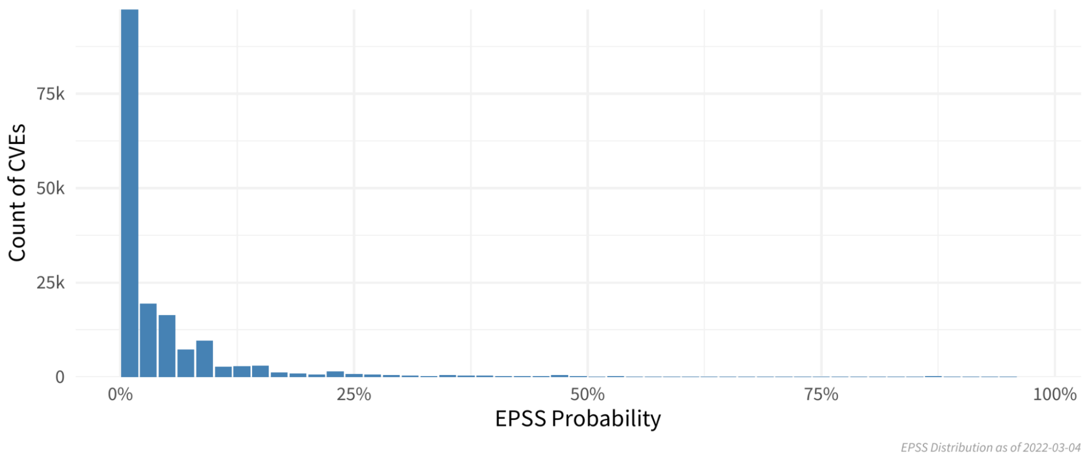


Figure 2.1: Distribution of EPSS probabilities as of 2022-03-04 (FIRST, 2022).

[19]https://www.cisa.gov/known-exploited-vulnerabilities-catalog
[20]https://googleprojectzero.blogspot.com/
[21]https://www.zerodayinitiative.com/
[22]https://www.exploit-db.com/
[23]https://github.com/
[24]https://www.metasploit.com/
[25]https://core.intrigue.io/
[26]https://sn1persecurity.com/wordpress/

The visualized distribution highlights the role of EPSS in guiding remediation strategies. By focusing on high-probability vulnerabilities, organizations can efficiently allocate their resources to address the most critical threats, reducing the likelihood of exploitation while maintaining operational security.

Within this thesis, the `first.org` EPSS database is queried via the provided API to retrieve exploit probability scores for each identified vulnerability. These scores are integrated into the multidimensional classification model, combining them with CVSS technical severity to enhance prioritization accuracy (see section 4.2).

2.3.5 Other Vulnerability Databases

While databases such as the Debian Security Tracker (DST)[27] and the Red Hat CVE Database (RHD)[28] also offer valuable insights for vulnerability management, they are not examined in detail in this thesis due to their limited scope and specific focus on particular software ecosystems. These databases primarily target vulnerabilities within their respective platforms - Debian-based and Red Hat systems - which can limit their general applicability across a wider range of software environments. The primary focus of this work is on vulnerability databases that provide broader, cross-platform coverage and have higher adoption rates across diverse development ecosystems. As such, databases like the NVD, OSV, and GAD were selected for their comprehensive, ecosystem-agnostic data.

2.4 Scientific Foundations of the Expert Questionnaire

The expert evaluation conducted within this thesis can be understood as a form of self-report methodology, as it depends on subjective judgments provided by experts based on their professional knowledge and experience. According to Lucas, Richard E. and Baird, Brendan M. (2006), self-report methodologies require respondents to interpret questions, recall relevant information, form judgments, and convert these judgments into responses. These cognitive processes underline the importance of careful questionnaire design, as each step can be substantially influenced by question wording, format, and context, thereby affecting the validity of responses (Lucas, Richard E. & Baird, Brendan M., 2006).

Further, as Schwarz (1999) notes, respondents infer the pragmatic meaning of questions using contextual clues, including explicit framing provided by the questionnaire. Clearly specifying the context of the questionnaire as an expert evaluation conducted within a master's thesis on cybersecurity can thus guide respond-

[27] https://security-tracker.debian.org/tracker/
[28] https://access.redhat.com/security/security-updates/cve

ents toward relevant and meaningful responses, enhancing the methodological rigor and reliability of the results.

Therefore, careful attention to question wording, context specification, and clear framing is essential for ensuring the validity of the expert evaluation, which this work has explicitly considered in designing the structured questionnaire.

The structured questionnaire used for the evaluation is presented in Appendix A.

2.5 Lessons Learned from Existing Tools and Their Impact on the Proposed Approach

The literature review of current vulnerability management tools (refer to section 2.2) shows that leading solutions adopt a multidimensional approach to vulnerability detection, assessment, and remediation. This approach is essential because vulnerabilities vary significantly in their characteristics and impacts, necessitating a comprehensive framework to effectively manage them. The following summary consolidates the scoring factors employed by prominent tools such as Tenable, Rapid7, and Snyk into a concise and comparative overview. By highlighting their respective strengths and specific scoring dimensions, this overview informs the development of a robust, hybrid vulnerability management model.

Consolidated Scoring Factors

Scoring Factor	Tenable	Rapid7	Snyk
Technical Severity (CVSS)	Yes	Yes	Yes
Exploit Availability & Maturity	Yes (VPR)	Yes (honeypot data)	Yes
Threat Intelligence	Yes	Yes	No
Business Context & Asset Criticality	Yes	Yes	No
Exposure	No	Yes	No
Fix Availability & Vulnerability Age	Yes (vulnerability age)	No	Yes (patch availability)
Additional Data Sources	Yes (internal DB)	Yes (partner data)	Yes (public/internal DB)

Table 2.1: Consolidated Scoring Factors used by Tenable, Rapid7, and Snyk

Summary and Impact on the Proposed Approach

Insights from Tenable, Rapid7, and Snyk highlight the need for a multidimensional approach to vulnerability management that integrates technical severity with real-world exploit likelihood. A notable limitation of relying solely on CVSS is scoring inconsistency, which can cause misaligned remediation priorities (J. Spring et al., 2021).

Integrating the quantitative severity assessment provided by CVSS with the probabilistic predictions from EPSS into a unified scoring system addresses this issue. This hybrid model incorporates factors such as vendor intelligence, vulnerability age, exploit availability, exploit maturity, and threat intelligence, which are dimensions successfully utilized by Tenable, Rapid7, and Snyk (see section 2.5). The studies conducted by Jacobs et al. (2021) and FIRST (2021) confirm that incorporating threat-likelihood metrics alongside CVSS improves remediation efficiency by reducing patch workloads while effectively mitigating actively exploited vulnerabilities.

Using EPSS and CVSS together significantly enhances prioritization: CVSS quantifies the *impact* of vulnerabilities, while EPSS estimates their *likelihood of exploitation* based on empirical data.

Figure 2.2 illustrates this concept clearly, showing the relationship between EPSS probabilities and CVSS scores for vulnerabilities as of 2021-05-16.

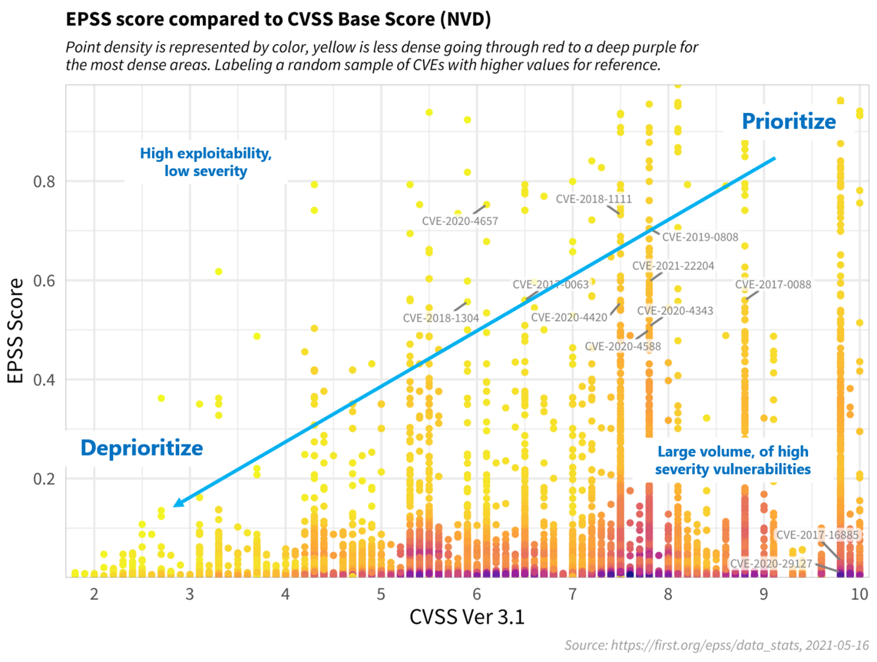


Figure 2.2: Illustration of how EPSS probabilities and CVSS severity scores complement each other for better risk-based prioritization (FIRST, 2021).

Most vulnerabilities cluster towards lower exploit probabilities. Only a small fraction of vulnerabilities have EPSS scores above 0.5. Although there is some correlation between EPSS and CVSS scores, this visualization clearly indicates that attackers do not exclusively target vulnerabilities that produce the greatest impact or are necessarily the easiest to exploit.

For prioritization, vulnerabilities in the lower-left quadrant (low probability, low impact) can typically be deprioritized. Vulnerabilities in the upper-left quadrant (high exploit probability, low impact) should be evaluated further, particularly in chained attack scenarios. Those in the lower-right quadrant (low exploit probability, high impact) warrant monitoring due to potential future exploitability. Vulnerabilities in the upper-right quadrant represent the highest risks (high probability, high impact) and must be addressed first (FIRST, 2021).

After establishing this combined severity score, the thesis applies a decision tree for remediation planning, drawing on the SSVC framework (see section 2.1.4). This tree incorporates *Vulnerability Classification*, *Patch Availability*, *System Usage* (public or internal), *Asset Criticality*, *Fix Complexity*, *Personal Code Ownership*, *Exploit Likelihood*, *Observed Exploits*, *Compliance Requirement*, and *Business Impact*. The required input data for these parameters is collected through a user query in the frontend, ensuring that context-specific factors are considered in the decision-making process.

By blending a robust quantitative metric with a well-defined remediation process, the proposed framework addresses all of the key data considerations employed by Tenable, Rapid7, and Snyk - ensuring precise prioritization and more informed remediation strategies for critical vulnerabilities.

3 Requirements

This chapter defines the requirements for the multidimensional vulnerability classification and remediation system developed in the context of this master's thesis. Requirements are clearly structured and categorized into functional and non-functional requirements.

3.1 Functional Requirements

The functional requirements specify the necessary capabilities and behaviors the system must deliver to users.

1. **Multidimensional Vulnerability Classification Model:** The system shall provide a model capable of classifying vulnerabilities across multiple relevant dimensions, considering technical and empirical factors.
2. **Algorithm for Computing Vulnerability Classifications:** The system shall include an algorithm that computes a severity classification for vulnerabilities based on relevant and available data sources.
3. **Remediation Recommendation Algorithm:** The system shall provide an algorithm to generate tailored recommendations for vulnerability remediation, taking into account factors such as stakeholder roles, asset importance, and patch availability.
4. **Rank-Ordering Model for Vulnerabilities:** The system shall provide a ranking model that prioritizes vulnerabilities according to their calculated severity, ensuring critical vulnerabilities receive immediate attention.
5. **Interactive Score Explanation Interface:** The system's user interface shall offer an interactive component that clearly explains how each vulnerability's overall severity was determined from underlying metrics.
6. **Handling of Missing Vulnerability Data:** When critical data required for scoring is unavailable, the system shall assign a placeholder score and clearly mark such vulnerabilities as having unknown severity.

7. **User Interface Warning for Missing Data:** The system shall explicitly inform users via the interface when necessary data is missing.
8. **Role-Based Decision Trees for Remediation:** The system shall incorporate structured decision trees to provide customized remediation recommendations tailored to different stakeholder roles.
9. **Role-Specific Recommendations:** The system shall support various stakeholder roles, such as developers and security coordinators, by providing recommendations relevant to their specific responsibilities.
10. **Caching Mechanism for External Data:** The system shall implement caching for external vulnerability data to reduce unnecessary network requests and ensure efficient data retrieval.
11. **Regular Data Synchronization:** The system shall perform regular synchronization with external vulnerability databases, maintaining updated and accurate vulnerability information.

3.2 Non-Functional Requirements

The non-functional requirements define quality criteria, constraints, and operational guidelines that the system must fulfill.

1. **Modularity:** The system architecture shall be modular, distinctly separating functions such as data management, scoring computation, caching, and interface interaction to enhance maintainability.
2. **Scalability:** The system must be scalable to accommodate increased data volumes and user interactions without performance degradation.
3. **Usability:** The user interface shall be intuitive, user-friendly, and provide clear guidance to ensure efficient and effective user interaction.
4. **Compliance with External API Rate Limits:** The system must manage external API requests responsibly, using strategies such as batching and caching to comply with rate limits and handle errors gracefully.
5. **Reliability and Error Handling:** Robust error handling mechanisms shall be implemented to maintain reliable system operation and clearly communicate any issues or missing information to users.
6. **Maintainability:** The system design shall follow clear separation of concerns, facilitating straightforward modifications, updates, or integration of additional functionalities.

The functional and non-functional requirements presented in this chapter collectively define the essential attributes and capabilities necessary to build a robust and effective system for vulnerability management, emphasizing key aspects such as modularity, scalability, usability, and transparency.

4 Multidimensional Vulnerability Classification and Remediation Framework

This chapter introduces a multi-dimensional framework for classifying and remediating vulnerabilities by integrating factors such as exploit predictability, vulnerability age, and attack complexity. Drawing directly from the conclusions of the literature review (chapter 2) and the summarized insights in section 2.5, the proposed approach addresses current limitations of the SCA Tool developed by the FAU Professorship for Open Source Software (OSS), which classifies vulnerabilities exclusively based on CVSS scores (Nehrke, 2023). Known limitations of this approach include inconsistent scoring among security professionals and its lack of empirical data on exploitability (Jacobs et al., 2021; J. Spring et al., 2021). To enhance vulnerability classification, the proposed framework combines CVSS severity scores with empirical exploit probability data provided by EPSS. For remediation prioritization, the stakeholder-specific SSVC decision framework is employed, leveraging context-sensitive factors including asset criticality, fix complexity, and personal code ownership.

The chapter covers:

- A model for multi-dimensional classification that evaluates vulnerabilities across multiple factors.
- An algorithm for computing classifications in specific software contexts.
- An algorithm for remediation recommendations tailored to stakeholders and asset criticality.
- A ranking model to prioritize vulnerabilities efficiently.

This framework aligns vulnerability management with both immediate and strategic security goals, addressing the complexities of modern software environments.

4.1 A Model for Multi-Dimensional Classification of Vulnerabilities

The following multi-dimensional classification model integrates several key dimensions to address both technical severity and real-world implications. Each dimension contributes to a holistic severity assessment; the dimensions used are:

- CVSS Score: Assesses technical severity using Version 3.1, which refines metrics and standardizes guidelines for comparability across systems (see section 2.1.2). It considers factors like the attack vector, complexity, and the potential impacts on confidentiality, integrity, and availability.
- EPSS Score: Represents the likelihood of real-world exploitation, based on empirical threat intelligence, such as publicly available exploits and observed attack trends.

As outlined in section 2.5, CVSS alone has been criticized for overlooking contextual factors and exhibiting scoring inconsistencies of 2–4 points among practitioners (J. Spring et al., 2021). To address these gaps, EPSS is incorporated as a complementary metric, leveraging real-world exploitation data to provide a more dynamic, threat-focused perspective. By combining CVSS's standardized impact assessment with EPSS's empirical likelihood estimations, this approach facilitates more context-aware remediation prioritization and aligns with industry best practices (see section 2.2). This synergy ensures that vulnerabilities with both high severity and a proven likelihood of exploitation receive immediate attention, while those posing lower real-world risk can be deprioritized, leading to more efficient allocation of remediation resources.

These dimensions form the foundation of the classification model used to evaluate vulnerabilities. By integrating standardized technical scoring with real-world exploitability data, this approach balances static severity metrics with dynamic threat intelligence and ensures that both long-term structural risks and immediate threats are considered. However, some organizations may need to factor in additional internal criteria - such as extended liability or compliance requirements - that lie beyond the scope of CVSS and EPSS.

4.2 Algorithm for Computing Classifications

The classification algorithm prioritizes vulnerabilities by combining these dimensions, resulting in a tailored severity score that reflects both technical severity and real-world implications:

1. *Input Retrieval:* Gather data for each vulnerability, including CVSS and EPSS scores.

2. *Score Calculation:* Combine standardized technical scores CVSS with real-world exploitability data EPSS by multiplying the EPSS probability value by 10.0 to align it with the 0–10 CVSS range. The overall severity score is then obtained using the following weighted formula:

$$\text{Severity Score} = \text{round}\left(\frac{w_{\text{cvss}} \times \text{CVSS} + w_{\text{epss}} \times \big(10.0 \times \text{EPSS}\big)}{2.0}\right)$$

   where each weight w is configured to reflect organizational priorities or estimations by domain experts.

3. *Classification and Output:* Each vulnerability is assigned a classification (e.g., Critical, High, Medium, or Low) based on the calculated score. This final output serves as guidance for prioritizing vulnerabilities and improving the overall security posture.

This work adopts a weighting factor of 2 for EPSS, deliberately emphasizing empirical exploit probability. Because the calculation of EPSS scores inherently incorporates certain elements from CVSS base metrics - such as exploitability characteristics - assigning a higher weight to EPSS emphasizes real-world exploitation likelihood while still effectively capturing the underlying technical severity. This approach ensures that vulnerabilities most likely to be actively exploited are prioritized, aligning vulnerability management closely with empirical evidence rather than purely theoretical severity assessments.

4.2.1 Data Sources for the Classification Model

The model integrates data from multiple sources to comprehensively evaluate vulnerabilities. Each vulnerability is referenced through a standardized CVE identifier, enabling consistent data retrieval. Additional metrics, such as technical severity from CVSS and empirical exploitation likelihood from EPSS, are obtained through this identifier. The primary data sources include the NVD, OSV, GAD, and the `first.org` EPSS database.

4.2.2 Handling Of Missing Data

If the data required for calculating the severity score (see section 4.2) is missing, the algorithm assigns a default value of **10.1**. This placeholder score is not displayed to the user. Instead, the frontend labels the severity in pink as "UNKNOWN", indicating that the severity could not be calculated. Since the maximum severity score is 10.0, a placeholder value of **10.1** ensures that the vulnerability is listed first on the dashboard.

4.2.3 User Interface for Explaining Score Calculation and Guidance in Case Of Missing Data

To enhance the user's understanding of the vulnerability scoring and prioritization, the system includes an interactive explanation feature and a method for handling incomplete data:

Score Details Button: The "Show Score-Details" button in the user interface provides detailed information about the calculated score. When clicked, it displays the following details:

- **CVE ID:** Shows the *Common Vulnerabilities and Exposures ID*, a unique identifier for the vulnerability. It helps security professionals track and reference the issue across different platforms and databases.
- **CVSS Score (v3.1):** Displays the *Common Vulnerability Scoring System* score, representing the severity of the vulnerability on a scale from 0.0 to 10.0. Higher scores indicate more critical vulnerabilities.
- **CVSS Vector:** Provides the *CVSS* vector string, which describes the characteristics of the vulnerability, such as attack vector, attack complexity, and privileges required for exploitation.
- **EPSS Score:** Shows the *Exploit Prediction Scoring System* score, which estimates the likelihood of the vulnerability being exploited in the wild.
- **Severity Score:** Displays the overall severity score, calculated based on the combination of the *CVSS* Score and *EPSS* Score. This score helps prioritize remediation efforts effectively.
- **Hover Tooltips for Vectors:** Displays hover-based tooltips for each *CVSS* vector attribute. When the user places the cursor over a specific field (e.g., "Attack Vector" or "Attack Complexity"), a concise explanation appears to clarify its impact on the vulnerability assessment.
- **Direct Link to the CVSS Calculator:** A direct link to a prefilled *CVSS* calculator, ensuring that users can quickly review the existing *CVSS* metrics. This link is automatically populated with the relevant vector data, allowing rapid exploration of different scoring scenarios.

Guidance in Case Of Missing Data: For cases where specific scores or recommendations are unavailable, the system transparently indicates missing components to the user. If critical data, such as the severity score, cannot be computed due to missing EPSS or CVSS scores, the user interface displays a clear red warning message stating: *"Cannot compute: missing EPSS or CVSS Score! Remediation Strategy: Please run the SSVC Assignment for further instructions!"* Additionally, the score details section may show placeholders like "CVSS: N/A" to inform users that certain information was not included in the score calculation, thus ensuring transparency.

This approach ensures that users are fully informed about how each vulnerability score is calculated, providing transparency and clarity. Additionally, handling missing data with fallback messages maintains consistency in user experience and supports informed decision-making, even when data is incomplete.

4.3 A Model for Remediation Recommendation

This section presents the SSVC vulnerability prioritization methodology, which leverages the SSVC framework to generate tailored remediation recommendations through structured decision trees. These decision trees guide stakeholders, such as developers and security advisors, in determining the most appropriate remediation actions for each vulnerability.

The SSVC framework, as discussed in section 2.1.4, does not directly affect the vulnerability score but guides remediation by aligning recommendations with organizational priorities. As outlined in section 2.5, it mirrors industry solutions (e.g., Tenable, Rapid7, Snyk) by factoring in elements such as *Exploit Status*, *Exposure*, *Asset Criticality*, and *Patch Availability*. The SSVC-based decision tree incorporates these parameters, gathered via a frontend user query, to ensure context-informed remediation.

Using the decision trees, vulnerabilities are categorized and prioritized for immediate patching, monitoring, or deprioritization. An example of such a decision tree tailored specifically to developers is illustrated in figure 4.1. For instance, if the exploit likelihood is high and a patch is available, developers should apply the patch immediately. If no patch is available, regular monitoring is recommended. Conversely, vulnerabilities with low exploit likelihood affecting non-critical assets can be deprioritized.

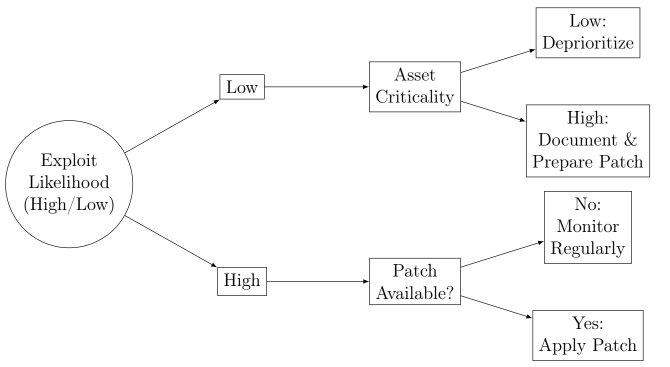


Figure 4.1: Example of an SSVC Decision Tree for Remediation Recommendations for the Developer Role

By applying this structured approach, the SSVC methodology ensures that remediation decisions align closely with stakeholder priorities and operational context, improving resource allocation and enabling more effective management of vulnerabilities.

4.4 Algorithm for Remediation Recommendation

This algorithm provides remediation recommendations based on the SSVC model. By considering factors such as stakeholder roles (e.g., developers, security coordinators), *Patch Availability*, and *Asset Criticality*, the algorithm generates tailored actions to address vulnerabilities efficiently. The following steps outline the decision-making process for each vulnerability:

1. *Input Retrieval and Initial Assessment:* The algorithm collects essential information about each vulnerability (e.g., the data points listed below), though in practice additional parameters may also be gathered to capture further contextual details:
    - *Stakeholder Role* to identify the responsibilities and potential impact for each user type, e.g., developers, operators, or security coordinators.
    - *Asset Criticality* to determine the priority based on the business importance of the affected asset.
    - *Patch Availability* to inform the urgency and feasibility of the remediation.
2. *SSVC-Based Decision Tree for Remediation Action:* The algorithm uses SSVC-inspired decision trees tailored to the roles of different stakeholders. In this example, the stakeholders include developers and security coordinators, each of whom receives specific recommendations based on their roles and responsibilities. This ensures that each role receives appropriate remediation guidance based on the specific characteristics of the vulnerability. Example actions for each stakeholder include:
    - **Developers:**
        - **High Exploit Likelihood and Patch Available:** Apply patch immediately to prevent exploitation.
        - **High Exploit Likelihood and No Patch Available:** Document the vulnerability, monitor regularly, and prepare for a future patch.
        - **Low Exploit Likelihood and Critical Asset:** Prepare patch documentation; consider patching in the next scheduled mainten-

ance.

- **Security Coordinators:**
  - **High Exploit Likelihood and Critical Asset:** Initiate immediate response, notify relevant teams, and enforce monitoring.
  - **Low Exploit Likelihood and Non-Critical Asset:** Document vulnerability details and set a reminder for review in future security audits.
  - **Significant Stakeholder Impact:** Prepare backup and documentation for affected systems, even if exploit likelihood is low, to ensure preparedness.

3. *Classification of Recommended Actions:* Based on the outputs of the SSVC-based decision tree, each vulnerability is assigned a recommended action classification, such as:
   - **Immediate Patch**: High-priority vulnerabilities with available patches are recommended for immediate remediation.
   - **Monitor and Prepare Patch**: Vulnerabilities with no immediate fix, but high exploit likelihood, are recommended for regular monitoring and preparation for a patch.
   - **Deprioritize**: Low-priority vulnerabilities affecting non-critical assets and posing minimal exploit risk are deprioritized but documented for future reference.
4. *Output of Recommendations:* The algorithm generates the recommended remediation actions for each vulnerability, taking into account the specific roles and priorities of each stakeholder. The recommendation provides a comprehensive view of immediate actions, monitoring tasks, and deprioritized items, ensuring that resources are allocated effectively to mitigate high-risk vulnerabilities.

This algorithm enables context-sensitive and efficient vulnerability management by aligning recommended actions with the needs of different stakeholders, such as developers and security coordinators, as well as the operational importance of each asset. This ensures that the most critical vulnerabilities are addressed promptly, while lower-risk issues are monitored or deprioritized.

4.5 A Model to Rank-Order All Classified Vulnerabilities

This model provides a structured approach for ranking all known vulnerabilities based on their calculated scores, prioritizing those with the highest severity to ensure efficient allocation of resources. By leveraging the composite scores generated through the multi-dimensional classification model, this ranking mechanism enables organizations to address the most critical vulnerabilities first. The steps for implementing this ranking model are outlined as follows:

1. **Score Aggregation:** The total score for each vulnerability is calculated from the weighted dimensions EPSS and CVSS (see section 4.2). This score provides a unified measure of severity.
2. **Handling Missing Data (Default Score 10.1):** When essential data for scoring is unavailable, the algorithm assigns a placeholder score of **10.1** - exceeding the maximum valid severity of 10.0 (see section 4.2.2). In the user interface, this appears as a pink **"UNKNOWN"** label rather than the numeric score, ensuring it is displayed first in the dashboard and prompting immediate attention to gather the missing information.
3. **Sorting and Rank-Order Calculation:** All classified vulnerabilities are sorted in descending order of their total (or placeholder) score, with the highest scores - whether valid or placeholder - representing the most urgent cases.
4. **Resource Allocation Guidance:** Based on the ranked list, organizations can allocate resources toward the vulnerabilities that pose the highest risk. This enables a focused remediation effort, ensuring that the most severe or unknown-risk vulnerabilities are addressed first.
5. **Dynamic Re-Ranking Based on Score Changes:** If there are updates to any of the vulnerability scores - such as new threat intelligence or changes in asset criticality - the model recalculates and reorders the list to reflect the latest context, keeping the priority list up-to-date.

This ranking model complements the multi-dimensional classification and remediation framework by establishing a clear, actionable prioritization order. It ensures that both high-risk and data-incomplete vulnerabilities are addressed promptly, aligning with strategic security objectives and operational capacity.

5 Design

This chapter describes the current and proposed designs for managing vulnerability data, emphasizing scalability and efficient interaction between system components. Building on the multidimensional classification and remediation model outlined in chapter 4, the proposed extension translates these concepts into a scalable and practical system design.

5.1 Current Design

The current solution integrates data from the OSV database into the internal system for vulnerability management. This process ensures efficient synchronization, processing, and storage of vulnerability advisories, aligning these advisories with components specified in the SBOM to identify packages and components affected by vulnerabilities.

- **Data Synchronization:** The system initiates synchronization by checking for the availability of the vulnerability data archive (OSV) in a cloud-based storage system. This includes mechanisms to verify the freshness of the data (e.g., through metadata such as ETags) to avoid redundant processing. Only updated or newly added archives are retrieved and temporarily stored for processing. This process happens once a day.
- **Data Extraction and Processing:** Retrieved archives are decompressed, and individual advisories are parsed and processed. Each advisory is evaluated for updates since the last synchronization and is enriched with relevant details, such as identifiers, severity metrics, and associated packages.
  - **Change Detection:** Only advisories that have been modified since the previous synchronization are processed further.
  - **Mapping and Enrichment:** Relevant details, including affected components, vulnerability types, and risk metrics (e.g., CVSS), are extracted and structured for database integration.
- **Component Alignment:** Processed advisories are matched against the

components listed in the SBOM. This step identifies specific packages and software elements within the SBOM that are directly affected by vulnerabilities, enabling precise vulnerability reporting at the component level.

- **Database Update:** The structured advisories, enriched with mappings to specific components, are stored in the internal database. Synchronization metadata, such as timestamps and version identifiers, is updated to reflect the latest state of external data sources. When an API request is triggered, the relevant data is sent to the frontend, ensuring the user interface always reflects the most recent vulnerability information.

5.2 Extended Design

The extended design centers on a streamlined approach to vulnerability information management. It integrates multiple data sources (e.g., vulnerability databases, scoring providers) and applies caching to reduce unnecessary network requests. Its purpose is to unify critical metrics (such as CVSS, EPSS) into a consolidated view, as described in the multi-dimensional classification model (see section 4.1), while also supporting role-specific actions based on the remediation model outlined in section 4.3.

5.2.1 Backend Responsibilities

The backend is responsible for data retrieval, caching, and score computation, following the scoring algorithm described in section 4.2:

- **Multi-Source Lookup:** For each vulnerability, the system first queries the OSV database to retrieve both the CVE identifier and (if available) the CVSS vector, leveraging OSV's integration of the entire GAD.[1] If no CVSS vector is found there, the system sends a request to external sources (e.g., the NVD), thereby minimizing the number of lookups. Similarly, the system always retrieves EPSS data from the `first.org` database to ensure it remains up to date, as it is not included in the internal OSV cache.
- **Caching and Time-Based Retrieval:** The system initiates a daily update cycle by first attempting to retrieve CVSS vectors from its local OSV database. If unavailable, it sends individual requests to the NVD for the missing CVSS details, as batch queries for CVE IDs are not supported.[2] Simultaneously, batch requests are sent to the `first.org` database for EPSS scores. After retrieving the new information, the cache is refreshed and any derived metrics (e.g., severity) are recalculated.

[1]https://osv.dev/

[2]https://nvd.nist.gov/developers/vulnerabilities

- **Role-Specific Recommendations:** Based on user input (e.g., developer or security advisor), the system determines a recommended course of action (see section 4.3). The backend solely stores the SSVC recommendation, which might include immediate patching, hotfixes, or a scheduled approach, for later retrieval or review.

5.2.2 Frontend Responsibilities

The frontend is responsible for fetching data from the backend, displaying it to the user, and forwarding user inputs for storage. Key functionalities include:

- **View Scores and Vectors:** A dedicated interface fetches data from the backend and displays CVE identifiers, CVSS vectors, CVSS base scores, EPSS values, overall severity, and missing data, as described in section 4.2.3. Additionally, it is responsible for showing vulnerabilities ranked as described in section 4.5.
- **Perform Structured Assignments:** An interactive decision flow guides users (depending on their role) through a series of questions, ultimately generating a recommendation, as seen in the concept of section 4.3.
- **Send Updates to the Backend:** Once the recommendation is finalized, it is transmitted to the backend for persistence.

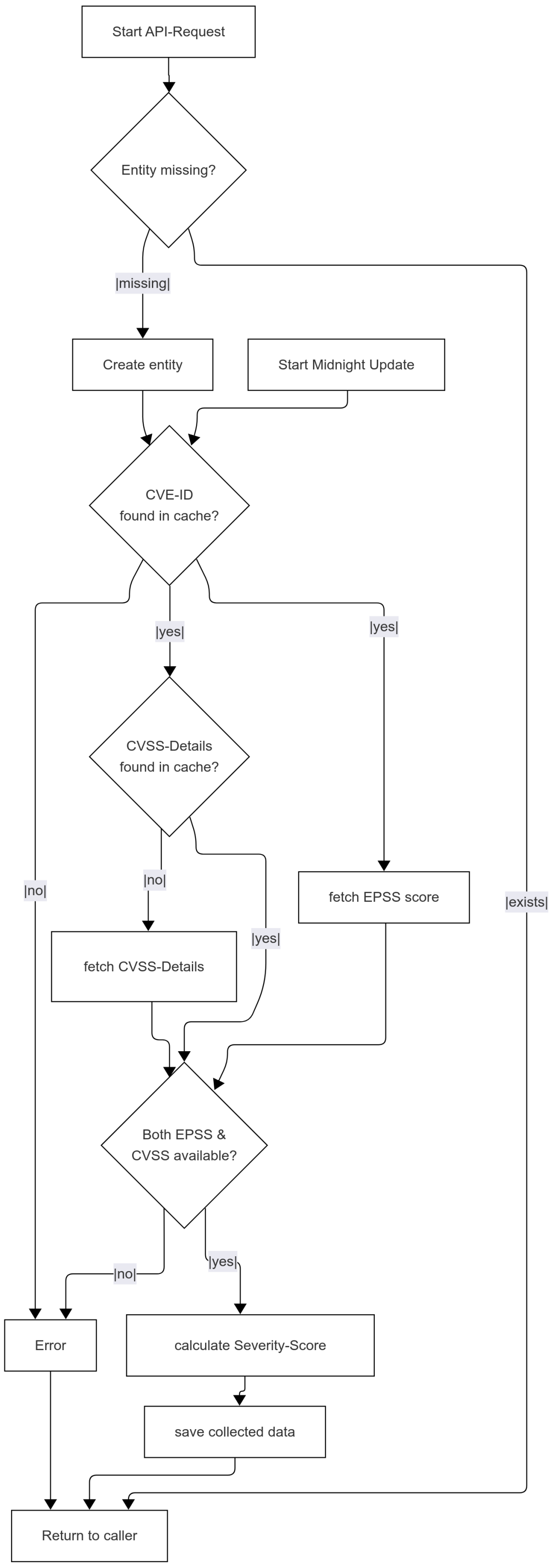
5.2.3 System Interaction Workflow

This section outlines the mechanisms for updating, fetching, and caching vulnerability data. Figure 5.1 shows how the process begins with a scheduled update (e.g., at midnight) or an API request from the frontend. In the latter case, if the vulnerability entity exists, the system returns it immediately; otherwise, a new record is created, and the creation or update process continues.

The backend checks if a CVE is in the cache. If not, the workflow transitions to an `Error` state, returning a severity score of **10.1** (see section 4.2.2), highlighted in pink on the frontend. Users can consult the *Score Explanation* feature (see section 4.2.3) for guidance. If the CVE exists, the system checks whether CVSS data is cached. If unavailable, an external fetch is performed. The EPSS metric is always retrieved externally to ensure it remains up to date, as it is not included in the cache.

Failure in either fetch leads to the `Error` state, while successful retrieval allows the system to compute an overall severity score (see section 4.2) and store the final data. The process concludes by returning a fully updated record to the caller.

Figure 5.1: The internal data-fetching, caching, and updating workflow for vulnerability data.



5.2.4 Rate Limits and Caching Benefits

In this design, CVE data is exclusively retrieved from the local OSV database, while CVSS data is primarily obtained from the local OSV database as well, querying the NVD only if the required information is unavailable locally. Additionally, EPSS data is fetched directly from `first.org`. External data sources impose rate limits to prevent misuse; for instance, the NVD restricts queries to five requests per 30-second window without an API key, and fifty with one.[3] The `first.org` database enforces a threshold of 1,000 requests per hour without a token.[4] Since CVE and most CVSS data are retrieved locally from the OSV database, no rate limits apply to these queries.

To cope with these constraints, the backend employs caching and batching to reduce repetitive lookups. Data retrieval occurs only when a record is missing or considered outdated, significantly minimizing external calls and reducing the risk of exceeding rate limits. During nightly updates, requests for EPSS scores are processed in batches to further optimize efficiency. However, since the NVD does not support batch queries for CVE IDs,[5] CVSS data retrieval requires individual requests, making caching even more essential. Consequently, the system operates more reliably and gracefully handles errors by returning a severity score of **10.1** (see section 4.2.2) whenever fetch errors occur. This condition is visually highlighted in pink on the frontend, prompting users to consult the score explanation interface (see section 4.2.3) to identify appropriate next steps for issue resolution.

5.2.5 SSVC Process and Role-Specific Assignments

The SSVC process involves tailoring vulnerability handling to each user's or team's specific context, following the stakeholder-specific prioritization framework described in section 4.4. This process is supported by the controller layer, which manages endpoints like the SSVC recommendation endpoint. These endpoints facilitate the submission of role-specific decisions, which are then persisted and integrated with other metrics (e.g., CVSS, EPSS). Figure 5.2 illustrates this interaction.

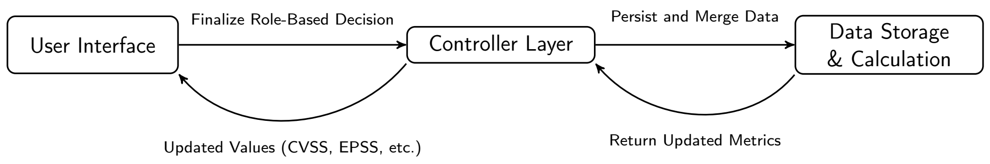

User Interface
Finalize Role-Based Decision
Controller Layer
Persist and Merge Data
Data Storage & Calculation
Updated Values (CVSS, EPSS, etc.)
Return Updated Metrics

Figure 5.2: High-level interaction in the SSVC process.

[3]https://nvd.nist.gov/developers/start-here
[4]https://api.first.org/#Rate-Limit
[5]https://nvd.nist.gov/developers/vulnerabilities

5.3 Conclusion

The outlined design consolidates key vulnerability metrics and merges them with role-based inputs to yield tailored recommendations. By systematically checking internal data first and fetching new information only when required, it minimizes redundant requests while maintaining data accuracy. Frontend interactions are kept straightforward through specialized dialogs and status views, ensuring users can easily view or update vulnerability details. This approach offers a strong foundation for integrating further data sources and scaling to large application ecosystems.

6 Implementation

This chapter focuses on the implementation of the components proposed in section 5.2. It describes how the backend functionalities, such as caching mechanisms, data acquisition workflows, and the SSVC recommendation process, were realized to ensure efficient data handling and seamless integration with external systems. Additionally, the chapter explains how the frontend was implemented to support user interactions, including role-specific remediation-strategies and data visualization. This implementation bridges the gap between architectural design and a functional, scalable system.

6.1 Backend Components

This section explains the main backend building blocks, focusing on data storage, retrieval logic, and the mechanisms used to ensure up-to-date information.

6.1.1 Caching Mechanism and Repository Layer

A central piece of functionality is the storage of previously retrieved data. Two core strategies are employed to manage data retrieval: **Pre-Check** and **Scheduled Refresh**.

The **Pre-Check** strategy involves querying the internal repository whenever a vulnerability request arises. The system first checks whether a corresponding *Vulnerability Data Entity* already exists. If it does, the existing record is returned. If no such entry is found, the system creates a new entity while simultaneously checking the local OSV database for the associated CVSS vector. If the vector is not available locally, an external API call to the NVD is performed to retrieve the missing details. In parallel, the EPSS score is fetched from the external `first.org` API, as it is not stored locally (see section 5.2.1).

The **Scheduled Refresh** strategy ensures data freshness through a nightly update cycle. This process starts 30 minutes after the OSV database update (see section 5.1), ensuring that the most recent vulnerability information is available.

During this cycle, the system updates the entire dataset by performing batch requests for EPSS scores via the `first.org` API and individual requests to the NVD for missing CVSS details not available in the local OSV database, as the NVD does not support batch queries for CVE IDs.[1]

If either the CVSS or EPSS data cannot be retrieved, the system transitions to an `Error` state, returning a severity score of **10.1**. This value is highlighted in pink on the frontend to prompt user action (see section 4.2.2).

Example of a Caching Strategy (Pseudocode)

```
// Caching Strategy Pseudocode

// Check if vulnerability exists
if (existsInRepository(cveId)) {
    return getFromRepository(cveId);
}

// Create new entity and fetch data
entity = createNewEntity(cveId);
entity.cvssVector = getFromOsv(cveId) ?? fetchFromNvd(cveId);
entity.epssScore  = fetchFromFirstOrg(cveId);

// Save and return entity
saveToRepository(entity);
return entity;
```

If any data retrieval fails, the process is aborted, and the system proceeds as described in section 4.2.2.

6.1.2 Scheduled Refresh Mechanism

The **Scheduled Refresh** strategy ensures data freshness through a nightly update cycle. This process starts 30 minutes after the OSV database update, ensuring that the most recent vulnerability information is available.

During this cycle, the system iterates through all existing *Vulnerability Data Entities* and updates each entry as follows:

- The local OSV database is queried for the latest CVSS vector.
- If the vector is not found locally, an external API call is made to the NVD.

[1]https://nvd.nist.gov/developers/vulnerabilities

- In parallel, the EPSS scores are fetched in a batch from the external `first.org` API, as it is not cached locally.

If any data retrieval fails, the system aborts the update for the affected entry and proceeds as described in section 4.2.2. Successfully updated entities are saved back to the repository, ensuring the dataset remains accurate and current.

Daily Vulnerability-Data Update (Pseudocode)

```
// Scheduled job: runs daily at 00:30
scheduleRecurringTask("vulnerability-data-update", "0 30 0 * * *") {
    updateVulnerabilityData();
}

function updateVulnerabilityData() {
    // Fetch all vulnerabilities
    entities = vulnerabilityDataRepository.findAll();

    // Batch-fetch EPSS scores
    epssScores = fetchBatchEpssScores(collectCveIds(entities));

    // Update each entity
    for each entity in entities {
        entity.epssScore  = epssScores.get(entity.cveId);
        [entity.cvssScore, entity.cvssVector] =
        fetchCvssScoreAndVector(
            entity.cveId,
            entity.vulnId
    );
        entity.severityScore = calculateSeverityScore(entity);
        entity.severityScoreLastUpdated = now();

        // Save updated entity
        vulnerabilityDataRepository.save(entity);
    }
}
```

SSVC Recommendation Endpoint The SSVC recommendation endpoint, part of the controller layer, allows the frontend to submit tailored remediation plans. As described in section 5.2.5, this endpoint integrates role-specific decisions into the backend's controller layer. These recommendations are stored in the backend for integration with other vulnerability metrics. The pseudo-implementation is as follows:

Updating SSVC Recommendation (Pseudocode)

```
// For a POST request to "/api/.../vulnerabilities/{vulnId}/ssvc"
function updateSsvc(vulnId, ssvcRecommendation):
    // Retrieve the vulnerability record by its ID
    data = repository.findById(vulnId)

    // If found, update recommendation and save
    if data:
        data.ssvcRecommendation = ssvcRecommendation
        repository.save(data)

    // Return success response
    return HTTP_200_OK
```

This process ensures that:

- SSVC recommendations are properly stored for each vulnerability.
- The data is integrated with other metrics such as CVSS and EPSS.
- The recommendations are available for retrieval and analysis in future requests.

6.1.3 Severity Score Calculation

After confirming the presence of CVSS, EPSS, and any other relevant metrics, the system generates a composite severity score. This calculation is based on the multi-dimensional classification model introduced in section 4.1, which integrates technical severity and real-world exploitability.

The algorithm follows the methodology described in section 4.2, prioritizing vulnerabilities by combining these metrics into a unified severity score. The specific steps are outlined as follows:

1. **Validate Required Fields**: If either the CVSS base score or EPSS value is missing, log an error and default them to **10.1**.
2. **Combine Weighted Values**: Multiply the EPSS probability by 10.0 to make it numerically compatible with the CVSS score (0–10).
3. **Compute Weighted Severity Score**: Calculate the severity score using the following pseudocode:

   ```
   double epssScaled = epss * 10.0;
   double severity = ((w_cvss * cvss) + (w_epss * epssScaled)) / 2.0;
   severity = round(severity);
   ```

Here, the weights w_{cvss} and w_{epss} follow the definitions provided in section 4.2, reflecting their relative importance in the overall severity assessment.

4. **Store the Result**: Update the database record with the newly calculated severity score.

Example of Combining CVSS and EPSS The following example demonstrates how CVSS and EPSS scores are combined to compute the severity score, aligning with the weighted scoring formula presented in section 4.2:

Severity Score Calculation (Pseudocode)

```
// Calculate severity score based on CVSS and EPSS
function calculateSeverityScore(data):
    cvss = data.getCvssScore()  // 0.0 to 10.0
    epss = data.getEpssScore()  // 0.0 to 1.0
    w_cvss = 1.0  // CVSS weight
    w_epss = 2.0  // EPSS weight

    if cvss is null or epss is null:
        return 10.1  // Error score if data is missing

    // Scale EPSS and calculate weighted average
    epssScaled = epss * 10.0
    finalScore = ((w_cvss * cvss) + (w_epss * epssScaled)) / 2.0

    return round(finalScore)
```

6.2 Frontend Components

This section highlights the primary elements of the frontend implementation, emphasizing user interface design, interaction logic, and the methods implemented to deliver a seamless user experience.

6.2.1 Role-Specific Logic and SSVC Integration

This section explains how the application adapts to different user roles, focusing on an SSVC-based approach as described in section 4.4 to provide tailored recommendations for remediation measures based on the user's specific role.

Interactive Decision Tree A component on the frontend prompts developers or security advisors through a sequence of questions (see figure 6.1). For instance, a developer might be asked:

- *Is a vendor patch already available?*
- *What is the level of criticality for your asset?*
- *Is this part of your personal codebase or third-party code?*

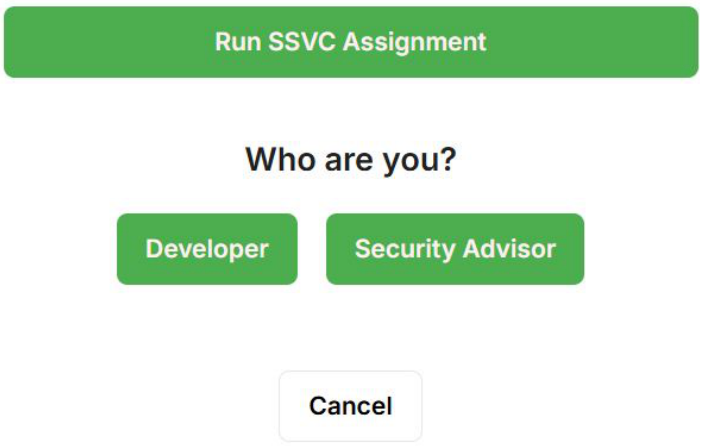


Figure 6.1: Interactive Decision Tree Interface: Initial role selection.

Each response is processed to build a comprehensive recommended action plan, consisting of advisories such as ‘Apply patch immediately’ when a vendor patch is available. Below is a code snippet demonstrating the logic for handling decisions within the tree structure and compiling the responses into a final recommendation. For example, one decision involves checking whether a vendor patch is available. Based on this, subsequent steps, such as applying the patch or exploring alternative mitigations, are determined.

Pseudocode for Decision Handling The following pseudocode demonstrates how decisions are processed to generate a recommendation:

```
// Generate recommendation based on patch availability
function generateRecommendation(data):
    recParts = []  // Collect recommendation parts

    if data.getPatchAvailableDev() == "yes":
        recParts.push("Vendor patch is available. Apply immediately.")
    else:
        recParts.push("No vendor patch found. Develop a custom fix.")

    return "Developer: " + recParts.join(" ")
```

Final Recommendation After collecting all partial recommendations, the final message is composed and delivered (see example figure 6.2):

```
// Compile and deliver final recommendation
finalMessage = "Developer: " + recParts.join(" ")
setSuccessMessage(finalMessage)
onComplete(finalMessage)
```

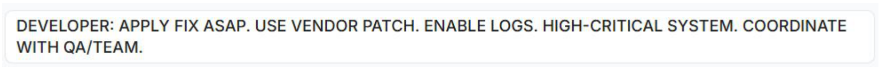

DEVELOPER: APPLY FIX ASAP. USE VENDOR PATCH. ENABLE LOGS. HIGH-CRITICAL SYSTEM. COORDINATE WITH QA/TEAM.

Figure 6.2: Example of a final recommendation message delivered to a developer.

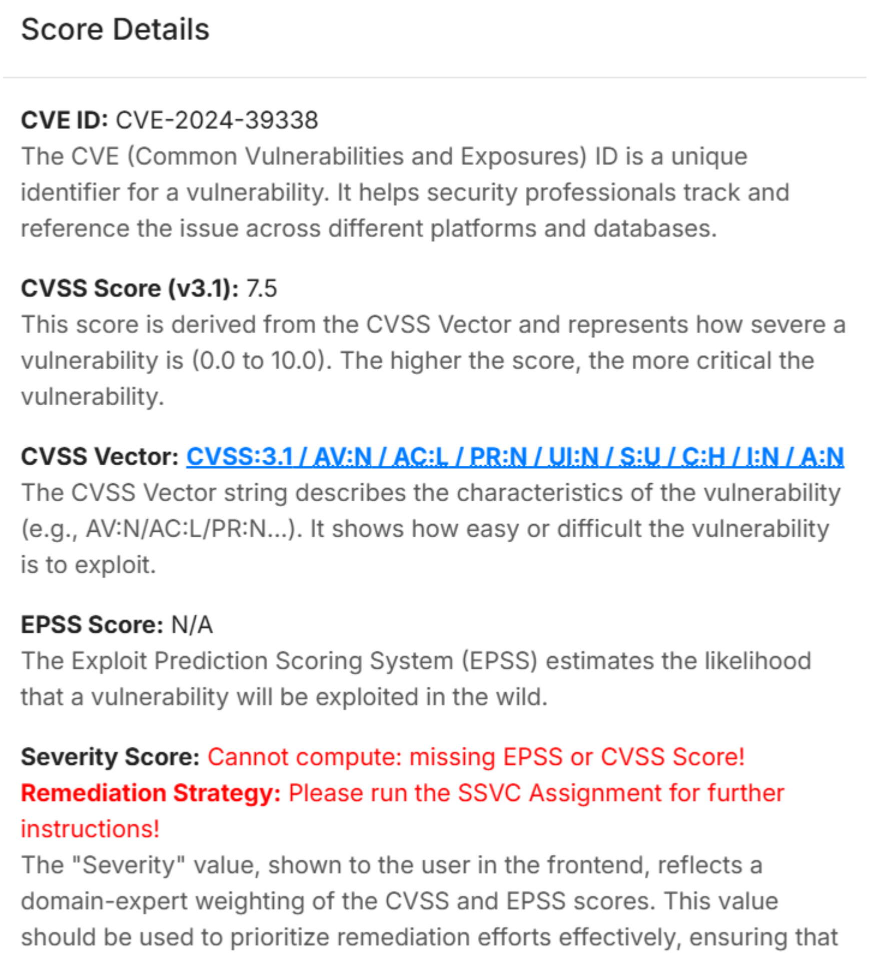
6.2.2 Score Details Button

A *Show Score Details* popup (as described in section 4.2.3) allows users to see the CVE-ID, the CVSS score and vector, the EPSS score, the computed severity, and a textual explanation as described in section 4.2.3. This is done through:

- **Button Trigger:** A button in the 'Vulnerability Details' view opens a modal dialog.
- **Dialog Contents:** The modal includes fields for the CVE ID, CVSS V3.1 base score, vector string, EPSS metric, and the final severity. If any fields are missing, placeholders (e.g., `N/A`) or a note about further actions that can be taken are shown.
- **Hover Tooltips for Vectors:** Displays hover-based tooltips for each CVSS vector attribute. When the user places the cursor over a specific field (e.g., 'Attack Vector' or 'Attack Complexity'), a concise explanation appears to clarify its impact on the vulnerability assessment.
- **Direct Link to the CVSS Calculator:** Provides a direct link to the official CVSS calculator.[2] This link is automatically populated with the relevant vector data, allowing users to quickly review the existing metrics and explore different scoring scenarios.

This functionality is illustrated in figure 6.3, which shows an example of the *Score Details* popup.

[2]https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator


Score Details

CVE ID: CVE-2024-39338
The CVE (Common Vulnerabilities and Exposures) ID is a unique identifier for a vulnerability. It helps security professionals track and reference the issue across different platforms and databases.

CVSS Score (v3.1): 7.5
This score is derived from the CVSS Vector and represents how severe a vulnerability is (0.0 to 10.0). The higher the score, the more critical the vulnerability.

CVSS Vector: CVSS:3.1 / AV:N / AC:L / PR:N / UI:N / S:U / C:H / I:N / A:N
The CVSS Vector string describes the characteristics of the vulnerability (e.g., AV:N/AC:L/PR:N...). It shows how easy or difficult the vulnerability is to exploit.

EPSS Score: N/A
The Exploit Prediction Scoring System (EPSS) estimates the likelihood that a vulnerability will be exploited in the wild.

Severity Score: Cannot compute: missing EPSS or CVSS Score!
Remediation Strategy: Please run the SSVC Assignment for further instructions!
The "Severity" value, shown to the user in the frontend, reflects a domain-expert weighting of the CVSS and EPSS scores. This value should be used to prioritize remediation efforts effectively, ensuring that

Figure 6.3: Example of the *Show Score Details* popup showing diverse data their explanation, and recommendations.

6.3 Ranking and Prioritization Of All Vulnerabilities

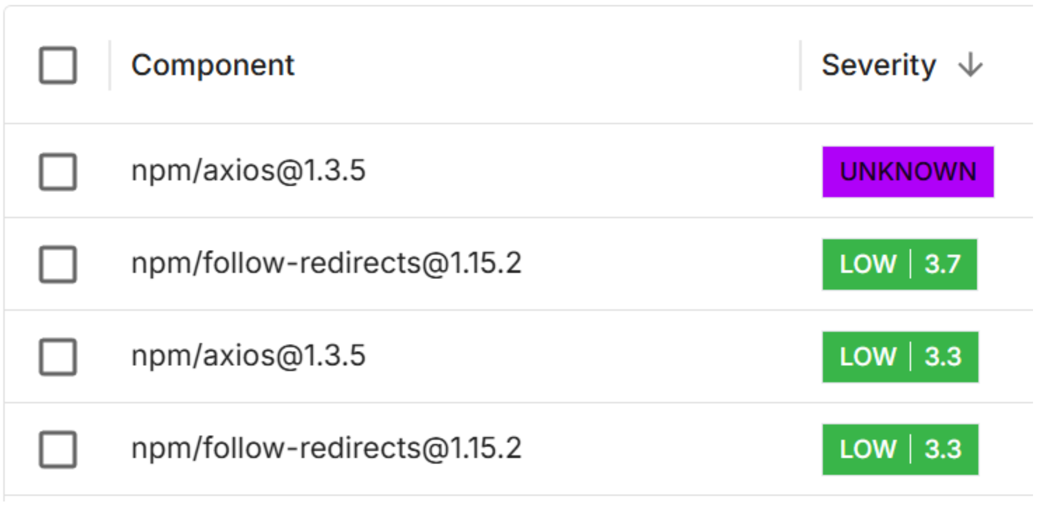
This section describes the implementation of the model introduced in section 4.5, focusing on how vulnerabilities are ranked and prioritized based on their severity.

6.3.1 Descending Sort of Precalculated Severity Scores and Handling Missing Data

The implementation takes the precalculated severity scores, which include contributions from CVSS and EPSS, and sorts them in descending order. This ensures that vulnerabilities with the highest severity scores are prioritized, while those with missing data are given placeholder scores for immediate attention.

For vulnerabilities with missing essential data, a placeholder score of **10.1** is assigned. This score exceeds the maximum valid severity of 10.0, ensuring these entries are sorted to the top of the list. In the user interface, such entries are highlighted with a pink **"UNKNOWN"** label, prompting users to investigate and resolve the missing information.

An example of the sorted list, as displayed in the user interface, is shown in figure 6.4, where placeholder scores for missing data are indicated by a pink 'UNKNOWN' label.



☐	Component	Severity ↓
☐	npm/axios@1.3.5	UNKNOWN
☐	npm/follow-redirects@1.15.2	LOW | 3.7
☐	npm/axios@1.3.5	LOW | 3.3
☐	npm/follow-redirects@1.15.2	LOW | 3.3

Figure 6.4: Example of a sorted list of vulnerabilities in descending order of severity.

6.4 Conclusion

The implementation details presented in this chapter show how the multi-dimensional vulnerability classification and remediation framework (see chapter 4) and the proposed system design (see chapter 5) have been realized in a fully functional solution that integrates both backend and frontend components. Critical elements, such as caching and scheduled refresh (see section 6.1.1), ensure that vulnerability data remains reliable, and continuously up-to-date, while minimizing external lookups.

The system interaction workflow (see section 5.2.3) highlights how local OSV data is leveraged, triggering external lookups for CVSS and EPSS only when strictly necessary. Building on these processes, the SSVC endpoint (see section 6.1.2) and role-specific frontend logic (see section 6.2) enable targeted remediation recommendations for different user roles.

Moreover, combining CVSS and EPSS metrics captures both technical severity and exploitability factors in accordance with the multi-dimensional model. Finally, the ranking and visualization (see section 6.3) allow for effective prioritization of vulnerabilities, with any missing data explicitly flagged via a placeholder severity score (e.g., **10.1**). Taken together, these components form a cohesive, user-focused system that significantly streamlines vulnerability management and remediation activities, fulfilling the objectives outlined in earlier chapters.

7 Evaluation

This chapter evaluates the implemented system described in chapter 6 against the functional and non-functional requirements defined in chapter 3, followed by an expert evaluation conducted through structured questionnaires to verify the system's applicability and effectiveness.

7.1 Evaluation of Functional Requirements

This section reviews each functional requirement, detailing how the implemented solution meets each criterion with specific examples.

1. **Multidimensional Vulnerability Classification Model (1)**

   The implementation provides a multidimensional vulnerability classification by combining multiple vulnerability metrics into a unified severity score. For example, as described in section 6.1.3, CVSS and EPSS metrics are combined to produce an actionable severity rating. **This requirement is fulfilled.**

2. **Algorithm for Computing Vulnerability Classifications (2)**

   The implemented classification algorithm calculates severity scores using available metrics (section 6.1.3). This includes combining a weighted CVSS score with a weighted EPSS score to provide a clear and precise classification. **This requirement is fulfilled.**

3. **Remediation Recommendation Algorithm (3)**

   The system generates tailored remediation recommendations using structured decision trees based on stakeholder roles, asset criticality, and patch availability (section 6.2.1). For instance, a developer is provided recommendations like 'Apply vendor patch immediately' if available or 'Develop a custom mitigation' otherwise. **This requirement is fulfilled.**

4. **Rank-Ordering Model for Vulnerabilities (4)**

   Vulnerabilities are ranked by their calculated severity, prioritizing the most critical vulnerabilities at the top (section 6.3). For example, vulnerabilities with higher computed severity scores appear prominently. **This requirement is fulfilled.**

5. **Interactive Score Explanation Interface (5)**

   The 'Show Score Details' interface displays comprehensive vulnerability information, including CVE identifiers, metrics, and an interactive, pre-filled link to the official CVSS calculator. Interactive tooltips offer detailed explanations for terms like 'Attack Vector' (section 6.2). **This requirement is fulfilled.**

6. **Handling of Missing Vulnerability Data (6)**

   When critical data is missing, the system assigns a placeholder severity score of 10.1, prioritizing the vulnerability at the top and visually highlighting it in pink with the label 'UNKNOWN'. This prompts users to address incomplete data entries (section 6.3). **This requirement is fulfilled.**

7. **User Interface Warning for Missing Data (7)**

   The user interface explicitly alerts users to missing data by highlighting vulnerabilities with a pink 'UNKNOWN' label, communicating the need for immediate corrective actions (section 6.3). **This requirement is fulfilled.**

8. **Role-Based Decision Trees for Remediation (8)**

   Interactive decision trees guide stakeholders through remediation options tailored to context. For instance, developers answering questions about patch availability and asset criticality receive actionable steps like immediate patching or monitoring (section 6.2.1). **This requirement is fulfilled.**

9. **Role-Specific Recommendations (9)**

   The system provides explicit role-specific recommendations. Developers receive instructions such as 'Apply patch immediately' (section 6.2.1). **This requirement is fulfilled.**

10. **Caching Mechanism for External Data (10)**

    Robust caching mechanisms significantly reduce external data retrieval. For instance, the system caches CVSS vectors locally and performs efficient batch retrieval of EPSS data (section 6.1.1). **This requirement is fulfilled.**

11. **Regular Data Synchronization (11)**

    A nightly synchronization cycle updates vulnerability data. Batch retrieval of EPSS data and individual queries to external databases (NVD) ensure data freshness, clearly illustrated by the scheduled refresh mechanism (section 6.1.2). **This requirement is fulfilled.**

7.2 Evaluation of Non-Functional Requirements

This section evaluates the non-functional requirements, emphasizing system quality, performance, and usability, and details how the implementation meets each requirement.

1. **Modularity (1)**

   The system distinctly separates backend functionalities (such as data caching, scheduled tasks, and repositories) from frontend interactions, facilitating maintainability. For instance, the repository layer manages data storage independently from the scoring algorithm and user interface logic (section 6.1). **This requirement is fulfilled.**

2. **Scalability (2)**

   Scalability is achieved through efficient batch-processing of external data (e.g., EPSS batch requests) and a scheduled refresh mechanism to handle growing data volumes and simultaneous user interactions without performance degradation (section 6.1.2). **This requirement is fulfilled.**

3. **Usability (3)**

   The system provides a user-friendly interface featuring clear navigation, interactive decision trees, and comprehensive score explanations. Interactive components, such as tooltips and pre-filled links to external calculators, significantly enhance usability (section 6.2). **This requirement is fulfilled.**

4. **Compliance with External API Rate Limits (4)**

   The implementation responsibly manages external API requests using caching and batching strategies, significantly reducing API calls and thus reducing rate-limit violations. (section 6.1.1). **This requirement is fulfilled.**

5. **Reliability and Error Handling (5)**

   Robust error handling mechanisms are implemented. For instance, when critical data (CVSS or EPSS) is missing, the system assigns a placeholder severity score of **10.1**, prioritizing the vulnerability at the top and visually marking it as ‘UNKNOWN’ to clearly communicate the issue (section 6.1.1). **This requirement is fulfilled.**

6. **Maintainability (6)**

   The modular architecture and clear separation of backend layers (controller, service, data layers) enable straightforward maintenance and integration of future enhancements, such as additional data sources or scoring mechanisms (section 6.1 and section 6.2). **This requirement is fulfilled.**

7.3 Evaluation of Models by Domain Experts

To further validate the practical relevance of the implemented framework, an expert evaluation was performed. Three domain experts from the field of cybersecurity assessed the applicability, clarity, and effectiveness of the models and algorithms presented in chapter 4. The evaluation was conducted using structured questionnaires provided in Appendix A and builds on the scientific foundations outlined in section 2.4. *Note: Expert statements have been partially paraphrased for clarity and brevity, while ensuring the original meaning was preserved.*

7.3.1 Multi-Dimensional Classification Model

Experts provided valuable feedback regarding the integration of CVSS and EPSS scores into a single unified severity score (see section 4.1). Expert feedback varied, with one expert explicitly stating: ‘I do not like merging CVSS and EPSS, as CVSS is already considered in EPSS according to page 2’. Another expert, however, emphasized the usefulness, stating: ‘The motivation to combine scores is clear and useful, although the selection of these two models specifically could be more clearly justified’.

Regarding weighting, suggestions included ‘a 50:50 ratio, initially not favoring one over the other,’ while another expert preferred a heavier weighting towards EPSS (70%) due to its empirical nature.

Overall expert ratings (1 = very good, 6 = insufficient):

Expert	Rating
Expert 1	3
Expert 2	1
Expert 3	2
Average	**2.0**

Table 7.1: Expert ratings for the Multi-Dimensional Classification Model

7.3.2 Algorithm to Compute Classification

The algorithm's (see section 4.2) handling of missing data through placeholder scores was generally perceived positively, although concerns were raised. One expert noted: 'The chosen placeholder could be confusing due to its proximity to valid score ranges; using infinity or NaN might be clearer'. Another expert commented positively: 'As long as it is clearly indicated, it should not confuse users'.

Experts also identified instances where the computed severity might differ from organizational practices, such as vulnerabilities in unused library components without valid threat vectors.

Overall expert ratings (1 = very good, 6 = insufficient):

Expert	Rating
Expert 1	2
Expert 2	1
Expert 3	2
Average	**1.67**

Table 7.2: Expert ratings for the Algorithm to Compute Classification

7.3.3 Remediation Mechanism

Experts confirmed that the SSVC-inspired remediation model and algorithm (see sections 4.3, 4.4) generally aligns with real-world practices but highlighted some deviations. One expert stated: 'Criticality of the asset is always considered, regardless of exploit likelihood'. Another emphasized: 'Even if the criticality is low, vulnerabilities must be monitored and patched, as a single flaw can become critical in an attack chain'.

Decision factors identified as most critical by experts were 'Exploit Likelihood and Asset Criticality, followed by Patch Availability'.

Overall expert ratings (1 = very good, 6 = insufficient):

Expert	Rating
Expert 1	2
Expert 2	1
Expert 3	2
Average	**1.67**

Table 7.3: Expert ratings for the Remediation Model (SSVC-inspired)

7.3.4 Rank-Ordering Mechanism

All experts agreed prioritizing vulnerabilities based on the highest scores or incomplete data (see section 4.5) matches standard practice. One expert suggested improvements: ‘Grouping vulnerabilities by technology stack or business unit might offer better practical usability’. Another proposed: ‘A second queue for unknown-data vulnerabilities could avoid blocking immediate mitigation tasks’.

Experts also suggested further grouping based on the type of vulnerability: ‘Software flaws need to be sorted and categorized by cyber hygiene (e.g., misconfigurations), compliance (driven by regulations), quality (software ’convenience’ patches), and security (software patches that fix vulnerabilities). All flaws need to be entered in a ticketing system and traced until solved’.

Overall expert ratings (1 = very good, 6 = insufficient):

Expert	Rating
Expert 1	3
Expert 2	1
Expert 3	2
Average	**2.0**

Table 7.4: Expert ratings for the Rank-Ordering Mechanism

7.3.5 Overall Expert Feedback and Recommendations

Overall, the expert evaluation indicated that the integrated models effectively reflect common vulnerability management practices, providing useful mechanisms for classification, remediation, and prioritization. Experts highlighted the practicality and clear alignment with current industry approaches, with one expert noting explicitly: ‘These integrated models reflect well how we manage open-source vulnerabilities in practice.’

Nevertheless, the evaluation identified areas for further improvement, particularly regarding clearer communication of placeholder scores, and the inclusion of compliance as a key cybersecurity driver: ‘Compliance is one of the drivers for cybersecurity and must always be considered.’ Additionally, experts suggested more granular grouping or filtering by technology stack or business units to enhance the usability of the rank-ordering mechanism. These considerations represent potential avenues for future enhancements of the framework.

7.4 Summary

This evaluation has demonstrated that the implemented system effectively meets all defined functional and non-functional requirements. The expert evaluation further validated the practical relevance and applicability of the multidimensional classification and remediation models, highlighting their alignment with common industry practices. Additionally, constructive feedback from domain experts identified opportunities for future enhancements, particularly concerning clearer communication of placeholder scores, improved grouping capabilities, and the integration of compliance considerations. Overall, the results confirm that the proposed solution provides a robust, scalable, and user-centric approach to vulnerability management.

8 Conclusion and Outlook

This chapter summarizes the contributions and findings of the thesis, highlighting its significance and limitations. Additionally, it outlines potential avenues for future research that could further enhance the proposed multidimensional vulnerability management framework.

8.1 Conclusion

This thesis proposed a multidimensional framework for the classification and remediation of vulnerabilities, aimed at addressing identified shortcomings within the existing vulnerability management component of SCA Tool developed by the OSS group at FAU. The existing approach, which relies exclusively on the CVSS, exhibits notable limitations, particularly inconsistencies among security practitioners' scoring and a lack of empirical exploitation data (J. Spring et al., 2021). To overcome these limitations, the proposed framework integrates technical severity metrics from CVSS with empirical exploit probability data provided by EPSS, thus offering a more holistic and context-aware method for vulnerability prioritization.

An evaluation against functional and non-functional requirements demonstrated that the proposed framework conceptually addresses all stated criteria, showing strong potential in enhancing vulnerability prioritization and remediation processes. Moreover, an expert evaluation confirmed the framework's practical relevance and its alignment with current industry standards and best practices. Experts highlighted particularly the advantage of integrating empirical exploitability measures alongside structured, stakeholder-specific remediation decision processes based on the SSVC framework.

However, the evaluation also revealed areas for further refinement. The method of combining CVSS and EPSS metrics elicited differing perspectives, with some experts noting conceptual overlaps that could potentially complicate interpretation. Additionally, the explicit integration of compliance aspects was recommended as a crucial extension to ensure alignment with organizational cybersecurity strategies.

In summary, the thesis conceptually advanced vulnerability management methodologies by proposing a comprehensive multidimensional classification and remediation framework. Its validation by domain experts underscores its potential effectiveness and practical applicability, while also highlighting avenues for further improvement.

8.2 Outlook

Several avenues for future research emerge from the presented framework and its evaluation:

Evaluation Against Ground Truth Data An important next step would be the empirical validation of the proposed classification framework against ground truth data. Evaluating the proposed model against datasets containing historically verified cases of exploited vulnerabilities would provide a robust empirical foundation for assessing the predictive accuracy and reliability of the integrated CVSS and EPSS classification approach.

Explicit Incorporation of Compliance Considerations Integrating compliance-driven factors, such as adherence to standards like General Data Protection Regulation (GDPR) or ISO 27001, into the vulnerability assessment and prioritization model would ensure a more holistic and strategically aligned approach. GDPR imposes strict regulatory obligations on organizations within the European Union (EU) to protect personal data (Chassang, 2017). Therefore, future research could explore explicitly identifying and prioritizing vulnerabilities affecting systems that process sensitive personal data, aligning vulnerability management more closely with these regulatory requirements. Similarly, the ISO 27001 standard provides a systematic framework for assessing and managing information security risks, including regular evaluation and prioritization of business-critical assets (International Organization for Standardization (ISO), 2018). Integrating ISO 27001 risk assessments into the proposed vulnerability classification framework would ensure that vulnerabilities affecting high-risk assets, as defined by organizational security policies, receive higher prioritization. Consequently, this explicit incorporation of compliance considerations would enhance the alignment between technical vulnerability management and broader organizational information security and regulatory strategies.

Advanced Grouping and Filtering Capabilities Lastly, introducing more granular grouping and filtering functionalities - such as by business units, technology stacks, or organizational responsibilities - could significantly enhance the practical usability of the framework. This would allow organizations to tailor vulnerability management more precisely to their specific operational contexts and strategic priorities.

These suggested research directions would contribute to refining the conceptual framework, ultimately fostering more effective and strategically aligned approaches to vulnerability classification and remediation.

Appendices

A Questionnaire: Evaluation of Models by Domain Experts

To ensure the practical relevance and effectiveness of the proposed models for vulnerability classification and remediation, an expert evaluation was conducted. The aim was to gather insights regarding the clarity, usability, and alignment of the models with real-world practices. Domain experts were invited to assess the following components:

- **Multi-Dimensional Classification Model:** This model integrates CVSS and EPSS scores to provide a comprehensive severity rating for software vulnerabilities. Experts evaluated the clarity and practicality of merging these scores and the appropriateness of the weighting approach.
- **Remediation Model (SSVC-inspired):** A decision-making framework providing tailored remediation recommendations based on exploit likelihood, patch availability, and asset criticality. The experts assessed whether the proposed decision flows align with current industry practices.
- **Rank-Ordering Mechanism:** A prioritization approach that highlights critical and data-incomplete vulnerabilities to streamline remediation efforts. Experts provided feedback on whether this prioritization reflects typical approaches used in practice.

The valuable feedback obtained from this evaluation directly influenced the refinement of the models, ensuring their applicability in real-world environments. It will also guide upcoming research in this area.

Note: The appendix contains the original questionnaire as an embedded PDF with separate pagination. The main thesis continues after the embedded document.

Master's Thesis: Evaluation of Proposed Models by Domain Experts

Felix Berger

March 7, 2025

Introduction and Purpose

This questionnaire is part of my master's thesis, in which I have developed four models for classifying and remediating software vulnerabilities. The following pages provide a concise overview of these models and invite feedback from domain experts. Your insights will directly influence how these models are refined, ensuring they are both theoretically sound and well-suited for real-world implementation. You can fill out the questionnaire **digitally** (if supported by your PDF reader) or print it and complete it manually.

The four models described here are:

- A multi-dimensional classification approach of vulnerabilities that integrates:
  - The **Common Vulnerability Scoring System (CVSS) v. 3.1**, which rates technical severity on a 0.0–10.0 scale.
  - The **Exploit Prediction Scoring System (EPSS)**, which estimates the probability (0.0–1.0) of a vulnerability being exploited in the wild.
- An algorithm to compute a classification for a known vulnerability in the context of a given software (a single numeric score, the *Combined Severity*).
- A remediation model offering different recommendations for roles (e.g., developers, security advisors).
- A rank-ordering mechanism that highlights critical or data-incomplete vulnerabilities first.

What CVSS Considers

CVSS provides a structured way to measure the severity of software vulnerabilities. Primary metrics include:

- **Attack Vector**: Network, adjacent, local, or physical.
- **Attack Complexity**: The conditions beyond the attacker's control that must exist to exploit a vulnerability.
- **Privileges Required**: The level of privileges an attacker needs.

- **User Interaction**: Whether user action is required for successful exploitation.
- **Scope**: Whether a vulnerability in one component impacts resources in another component.
- **Confidentiality, Integrity, Availability Impacts**: The potential effects on data or system.

What EPSS Considers

EPSS is an empirical model predicting how likely a vulnerability is to be exploited in practice. It draws upon:

- **Vendor Information**: Derived from CPE via NVD.
- **Age of the Vulnerability**: Days since the CVE was published by MITRE.
- **References**: With categorical labels, e.g., MITRE CVE List, NVD.
- **Normalized Multiword Expressions**: Extracted from vulnerability descriptions (MITRE CVE List).
- **Weakness Details**: CWE identifiers from NVD.
- **CVSS Metrics**: Base vectors from CVSS 3.x (via NVD).
- **Listings on Well-Known Sites**: CISA KEV, Google Project Zero, Trend Micro's ZDI.
- **Publicly Available Exploit Code**: Exploit-DB, GitHub, Metasploit.
- **Offensive Security Tools/Scanners**: Intrigue, sn1per, jaeles, nuclei.

1) Multi-Dimensional Classification Model

Objective. This model seeks to determine the most suitable approach for assigning a single severity score to software vulnerabilities. Instead of relying exclusively on a predefined metric (e.g., CVSS v. 3.1) or a single data source (e.g., EPSS), the model aims to integrate and balance multiple factors to produce a more nuanced assessment of vulnerability severity. By systematically refining these inputs and their relative weights, the model aspires to offer a flexible and context-aware classification method that can outperform any single, static metric.

Formula Example.

$$\text{Severity Score} = \text{round}\left(\frac{w_{\text{cvss}} \times \text{CVSS} + w_{\text{epss}} \times \big(10.0 \times \text{EPSS}\big)}{2.0}\right),$$

where each w represents the relative importance assigned to its respective factor, and round is a typical rounding function. Multiplying EPSS by 10 aligns it with CVSS's 0.0–10.0 scale.

Illustrative Calculation. If CVSS = 7.5, EPSS = 0.4, and both weights = 1.0:

$$\text{Combined Severity} = \text{round}\left(\frac{7.5 + (10 \times 0.4)}{2}\right) = \text{round}(5.75) = 6.$$

Depending on internal thresholds, that might be classified as "Medium" or "High."

2) Algorithm to Compute Classification

1. **Data Retrieval**: Gather CVSS and EPSS from sources (NVD, GitHub Advisory Database, OSV, etc.).
2. **Scaling**: Multiply EPSS by 10.
3. **Combination**: Apply the chosen formula to produce a single score.
4. **Fallback**: If key data is missing, assign a placeholder score. This placeholder score is not displayed to the user. Instead, the frontend labels the severity in pink as "UNKNOWN", indicating that the severity could not be calculated. Since the maximum severity score is 10, a placeholder value of 10.1 ensures that the vulnerability is listed first on the dashboard.

3) Remediation Model (SSVC-inspired)

This model generates actions such as "apply patch immediately," "monitor regularly," or "deprioritize," guided by role-based decision flows. Below is a short example of such a flow for a developer:

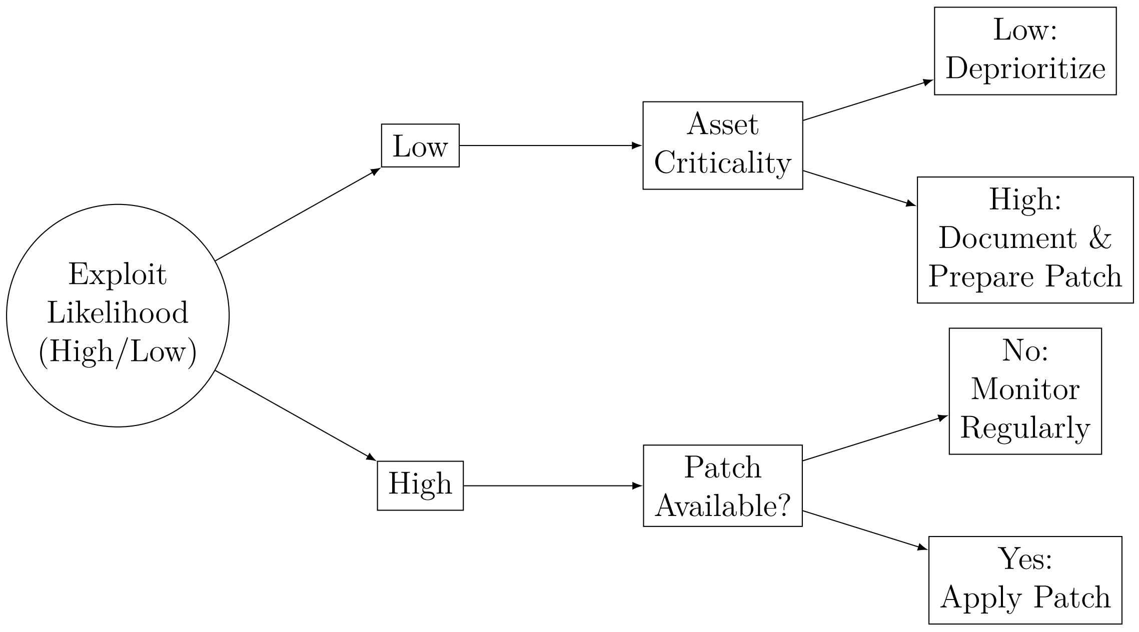


Figure 1: Abbreviated SSVC decision tree example (developer role).

Combined Outcomes (Sample Code Snippets).

- *Developer: Apply fix ASAP. Use vendor patch. High-critical system.*
- *Developer: Investigate further. No vendor patch. Coordinate with QA/team.*
- *Security-Advisor: Immediate patch. Active exploits detected. Ensure regulatory compliance.*

4) Rank-Ordering Mechanism

All vulnerabilities receive the final score (from Section 1 or 2). They are then sorted in descending order. If data is missing (*CVSS*, *EPSS*), a placeholder score (e.g., 10.1) elevates that entry to the top, prompting resolution of data gaps or running the SSVC assignment.

Questionnaire

Instructions: You can type into the text fields below (if supported by your PDF reader) or print and fill them by hand.

A) Multi-Dimensional Classification Model

1. **How clear or useful is it to merge CVSS and EPSS into a single severity score?**

2. **What approximate weightings (for general applications) would you choose for CVSS vs. EPSS, and why?**

3. **How would you rate the Multi-Dimensional Classification Model (1 = very good, 6 = insufficient)?**

B) Algorithm to Compute Classification

1. **Is the fallback (placeholder score) for missing data beneficial, or could it lead to confusion?**

2. **Please provide an instance where the computed severity might differ from your organization's internal approach or policy.**

3. **How would you rate the Algorithm to Compute Classification (1 = very good, 6 = insufficient)?**

C) Remediation Model (SSVC-inspired)

1. Do the (radically simplified) example SSVC decision flows generally align with how you handle vulnerabilities (e.g., developer vs. security advisor)?

2. **In the SSVC approach, which decision factors (e.g., Exploit Likelihood, Patch Availability, Asset Criticality) do you consider most critical for accurate**

remediation (for developers and security advisors)?

3. **How would you rate the Remediation Model (SSVC-inspired) (1 = very good, 6 = insufficient)?**

D) Rank-Ordering Mechanism

1. **Does placing highest-scored or unknown-data items on top match your typical prioritization approach?**

2. **Would grouping vulnerabilities by technology stack, business unit, or other factors be more practical?**

3. **How would you rate the Rank-Ordering Mechanism (1 = very good, 6 = insufficient)?**

E) Overall Perspective

1. **Do these integrated models (classification, remediation, rank-ordering) reflect how you would generally manage open-source vulnerabilities?**

2. **Are there any technical or organizational considerations missing that should be included?**

3. **Additional remarks or suggestions:**

Name, Role, and Signature

Organization: Capgemini Outsourcing Services GmbH

Name:

Role in Organization:

Date: **Signature:**

References (Questionnaire)

- FIRST.org (2024). “Exploit Prediction Scoring System.” `https://www.first.org/epss/`
- NIST (2024). “National Vulnerability Database.” `https://nvd.nist.gov/`
- GitHub (2024). “GitHub Advisory Database.” `https://github.com/github/advisory-database`
- OSV (2024). “Open Source Vulnerabilities Database.” `https://osv.dev/`

References

Balbix, Inc. (2020, May). CVSS v2 vs CVSS v3. Retrieved November 28, 2024, from https://www.balbix.com/insights/cvss-v2-vs-cvss-v3/

Black Duck Software, Inc. (2024). Open Source Security & Risk Analysis Report (OSSRA) | Black Duck. Retrieved November 28, 2024, from https://www.blackduck.com/resources/analyst-reports/open-source-security-risk-analysis.html

Chassang, G. (2017). The impact of the EU general data protection regulation on scientific research. *ecancermedicalscience*. https://doi.org/10.3332/ecancer.2017.709

FIRST. (2021, May). EPSS User Guide. Retrieved February 10, 2025, from https://www.first.org/epss/user-guide

FIRST. (2022). Exploit Prediction Scoring System (EPSS). Retrieved November 30, 2024, from https://www.first.org/epss/articles/prob_percentile_bins

FIRST. (2024a). CVSS v3.1 User Guide. Retrieved January 6, 2025, from https://www.first.org/cvss/v3.1/user-guide

FIRST. (2024b). CVSS v4.0 Specification Document. Retrieved November 28, 2024, from https://www.first.org/cvss/v4.0/specification-document

FIRST. (2024c). The EPSS Model. Retrieved October 27, 2024, from https://www.first.org/epss/model

FIRST. (2025). CVSS v3.1 Specification Document. Retrieved March 29, 2025, from https://www.first.org/cvss/v3-1/specification-document

GitHub, Inc. (2024). About the GitHub Advisory database. Retrieved October 28, 2024, from https://docs.github.com/en/code-security/security-advisories/working-with-global-security-advisories-from-the-github-advisory-database/about-the-github-advisory-database

International Organization for Standardization (ISO). (2018). *ISO/IEC 27000:2018 Information technology – Security techniques – Information security management systems – Overview and vocabulary* (tech. rep. No. ISO/IEC 27000:2018). International Organization for Standardization.

Jacobs, J., Romanosky, S., Edwards, B., Adjerid, I., & Roytman, M. (2021). Exploit Prediction Scoring System (EPSS). *Digital Threats: Research and Practice*, *2*(3), 1–17. https://doi.org/10.1145/3436242

Lucas, Richard E. & Baird, Brendan M. (2006). Global Self-Assessment. In Eid, Michael & Diener, Ed (Eds.), *Handbook of Multimethod Measurement in Psychology* (pp. 29–42). American Psychological Association.

Microsoft, C. (2024, August). Compare Microsoft Defender Vulnerability Management plans and capabilities - Microsoft Defender Vulnerability Management. Retrieved October 27, 2024, from https://learn.microsoft.com/en-us/defender-vulnerability-management/defender-vulnerability-management-capabilities

MITRE Corporation. (2024). Overview | CVE. Retrieved November 28, 2024, from https://www.cve.org/About/Overview

Nehrke, L. (2023). Webdienst zur überwachung von Schwachstellen in Software-Stücklisten (SBOM). https://oss.cs.fau.de/wp-content/uploads/2024/02/Nehrke_2023.pdf

NIST. (2024, June). NVD - CVEs and the NVD Process. Retrieved October 27, 2024, from https://nvd.nist.gov/general/cve-process

OSV. (2024). Introduction to OSV. Retrieved October 27, 2024, from https://google.github.io/osv.dev/

Qualys. (2022, June). Qualys führt VMDR 2.0 mit TruRisk™ Scores und automatisierten Remediation Workflows ein. Retrieved October 27, 2024, from https://www.qualys.com/company/newsroom/news-releases/germany/qualys-fuehrt-vmdr-2-0-mit-trurisk-tm-scores-und-automatisierten-remediation/

Rapid7. (2017, June). Live Threat-Driven Vulnerability Prioritization | Rapid7 Blog. Retrieved October 24, 2024, from https://www.rapid7.com/blog/post/2017/06/13/live-threat-driven-prioritization/

Schwarz, N. (1999). Self-reports: How the questions shape the answers [Place: US Publisher: American Psychological Association]. *American Psychologist*, *54*(2), 93–105. https://doi.org/10.1037/0003-066X.54.2.93

Snyk Limited. (2023). *2023 State of Open Source Security* (tech. rep.). Retrieved November 28, 2024, from https://go.snyk.io/state-of-open-source-security-report-2023-dwn-typ.html

Snyk Limited. (2024a). Die vier Schritte der Schwachstellenbehebung. Retrieved October 24, 2024, from https://snyk.io/de/learn/vulnerability-remediation-process/

Snyk Limited. (2024b, April). Snyk Vulnerability Database | Snyk User Docs. Retrieved October 24, 2024, from https://docs.snyk.io/scan-with-snyk/snyk-open-source/manage-vulnerabilities/snyk-vulnerability-database

Snyk Limited. (2024c, September). Priority Score | Snyk User Docs. Retrieved February 9, 2025, from https://docs.snyk.io/manage-risk/prioritize-issues-for-fixing/priority-score

Snyk Limited. (2024d, December). Risk Score | Snyk User Docs. Retrieved February 9, 2025, from https://docs.snyk.io/manage-risk/prioritize-issues-for-fixing/risk-score

Spring, J., Hatleback, E., Householder, A., Manion, A., & Shick, D. (2021). Time to Change the CVSS? *IEEE Security & Privacy*, *19*(2), 74–78. https://doi.org/10.1109/MSEC.2020.3044475

Spring, J. M., Hatleback, E., Householder, A. D., Manion, A., Oliver, M., Sarvapalli, V., Shick, D., & Tyzenhaus, L. (2021). Prioritizing vulnerability response: A stakeholder-specific vulnerability categorization (version 2.0).

Tenable, Inc. (2024a). Vulnerability Intelligence. Retrieved October 24, 2024, from https://docs.tenable.com/vulnerability-management/Content/vulnerability-intelligence/vuln-intel-intro.htm

Tenable, Inc. (2024b). Tenable Vulnerability Management User Guide, 43–46. Retrieved October 23, 2024, from https://docs.tenable.com/vulnerability-management/Content/PDF/Tenable_Vulnerability_Management-User_Guide.pdf