In-Depth Analysis of Software Composition Analysis Tools: Current Curation Practices

MASTER THESIS

Aleem Ud Din Aleem Ud Din

Submitted on 6 May 2025

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

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

Declaration of Originality

I confirm that the submitted thesis is original work and was written by me without further assistance. Appropriate credit has been given where reference has been made to the work of others. The thesis was not examined before, nor has it been published. The submitted electronic version of the thesis matches the printed version.

Erlangen, 6 May 2025

License

Erlangen, 6 May 2025

Acknowledgments

I would like to express my sincere gratitude to my primary supervisor, Mr. Martin Wagner, for his invaluable guidance, consistent support, and constructive feedback throughout the course of this thesis. His expertise and mentorship at the Open Source Research Group, Computer Science Department, Friedrich-Alexander University, were instrumental in shaping this work.

I also acknowledge Prof. Dr. Dirk Riehle for his formal supervision as the head of the research group.

I am thankful to the faculty and staff at Friedrich-Alexander University for providing a supportive academic environment.

My heartfelt thanks go to my family and friends for their continuous encouragement and understanding. I also made selective use of tools such as ChatGPT and Grammarly to assist with sentence structuring, refining grammar, and clarifying specific ideas during the writing process.

Abstract

Software Composition Analysis (SCA) tools play a vital role in identifying vulnerabilities and ensuring license compliance in Open Source Software (OSS). However, their effectiveness is strongly influenced by the quality of metadata used during analysis. Incomplete, inconsistent, or outdated metadata can lead to false positives, undetected vulnerabilities, or misclassified licenses. This thesis investigates how SCA tools address these challenges through metadata curation practices focused on validation, correction, and updating.

Drawing on a Systematic Literature Review (SLR) of 25 academic papers and selected grey literature, this study analyzes 31 SCA tools, including both full-featured solutions and supporting components (e.g., license classifiers such as Ninka). These tools are classified by their primary focus (vulnerability or licensing), curation methodology (manual, automatic, or hybrid), data sources, and update frequency. The findings reveal that while a few tools employ structured, community-driven curation workflows, most provide limited transparency regarding how metadata is curated or maintained.

The thesis proposes a typology of curation strategies, identifies critical gaps in transparency and evaluation, and draws comparative lessons from domains where metadata quality is a primary concern. These insights lead to practical recommendations for improving the reliability, auditability, and interpretability of SCA tools through more modular and transparent curation workflows.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

OSS Open Source Software

AST Abstract Syntax Tree

BSD-3-Clause Berkeley Software Distribution 3-Clause License

BDSAs Black Duck Security Advisories

CNA CVE Numbering Authority

CVE Common Vulnerabilities and Exposures

CVSS Common Vulnerability Scoring System

FOSS Free and Open Source Software

GHSA GitHub Security Advisories

GPL GNU General Public License

JSONL JSON Lines

LLM Large Language Model

NVD National Vulnerability Database

ODC OWASP Dependency-Check

OMP osslist-maven-plugin

OSV Open Source Vulnerabilities

OSS Index Sonatype Open Source Software Index

OSSRA Open Source Security and Risk Analysis

PII Personally Identifiable Information

SBOM Software Bill of Materials

SCA Software Composition Analysis

SLR Systematic Literature Review

SPDX Software Package Data Exchange

CI/CD Continuous Integration / Continuous Deployment

SDLC Software Development Lifecycle

API Application Programming Interface

PCFG Probabilistic Context-Free Grammar

SACG structure-aware call graphs

CyRC Cybersecurity Research Center

1 Introduction

OSS has significantly transformed the landscape of modern software development, empowering developers and organizations to rapidly innovate, reduce costs, and accelerate product delivery. However, this increased reliance on OSS has brought substantial risks, notably security vulnerabilities and challenges related to legal compliance. These risks primarily stem from the use of third-party software components, and are compounded by the fact that metadata, such as vulnerability reports and licensing information stored in public or proprietary databases, is often incomplete, outdated, or inaccurate.

The 2025 Open Source Security and Risk Analysis (OSSRA) report found that OSS is nearly universal in commercial software, with 97% of the audited applications containing open source components (OSSRA, 2025). The same report also revealed that 91% of codebases contained outdated open source components, often more than 10 versions behind, underscoring the ongoing risk associated with unmanaged dependency updates. Additionally, 86% of applications had known open source vulnerabilities, and 64% of the open source code originated from transitive dependencies, which are typically more difficult to monitor and secure. In addition to security risks, license conflicts were widespread: 56% of audited applications had license conflicts, and 33% contained components with either no license or a customized license. Notably, nearly 30% of license conflicts were caused by transitive dependencies, reinforcing the complexity and opacity of modern software supply chains.

To manage these risks, SCA tools have become indispensable. They help organizations systematically identify vulnerabilities, verify compliance with software licenses, and mitigate third-party risks in their software supply chains. Yet, despite their growing adoption, there is limited transparency regarding how these tools manage and curate the metadata underpinning their analysis. The internal processes used by SCA tools to validate, correct, and update critical metadata such as vulnerability identifiers and license compliance often remain opaque and inconsistently documented. Organizations relying on SCA tools frequently encounter reliability issues rooted in insufficient or inconsistent metadata curation. Poorly curated metadata can lead to false positives, misclassified licenses, and

conflicting outputs across tools, all of which undermine the accuracy and efficiency of vulnerability remediation. For example, 14.8% of organizations report frequent failures in remediation processes (Snyk & Foundation, 2024), highlighting the operational impact of unreliable or incomplete security data. Without robust validation and ongoing maintenance of automatically retrieved metadata, organizations risk unnecessary remediation efforts, legal missteps, and exposure to unaddressed vulnerabilities, ultimately compromising both security and compliance.

The consequences of poor metadata curation illustrate why it is crucial to examine how SCA tools manage and maintain the metadata they rely on. However, this aspect has received limited attention in existing research, and detailed documentation of curation practices is often lacking. To address this gap, this thesis investigates how current SCA tools approach metadata curation by reviewing available documentation and interpreting practices based on tool behavior and public sources. In addition, the thesis explores selected curation strategies from other domains as potential sources of inspiration for improving metadata handling in SCA tools, particularly with respect to consistency, validation, and community involvement.

By conducting a focused and structured analysis of these practices, this work aims to shed light on how SCA tools operate beyond their advertised capabilities. Through this transparency-focused and systematic approach, the research directly addresses the operational realities faced by software practitioners, helping stakeholders make informed decisions about selecting and trusting SCA tools. Ultimately, the findings contribute to improving metadata quality and reliability, thereby strengthening software security and compliance practices across organizations.

1.1 The Role and Significance of SCA Tools

SCA tools are designed to identify, evaluate, and manage open-source components in software projects. They play a vital role in reducing risks associated with software reuse, particularly by detecting security vulnerabilities (Dietrich et al., 2023), and ensuring license compliance.

Typically, SCA tools consist of a dependency scanner that inspects source code to detect third-party libraries. Detected components are then matched against known vulnerability and license databases. Notable examples include the National Vulnerability Database (NVD), as defined by NVD (n.d.), and the GitHub Security Advisories (GHSA), introduced by GHSA (n.d.), as well as vendor-maintained repositories or open standards like the Software Package Data Exchange (SPDX), a Linux Foundation project for standardizing metadata about software licenses

(SPDX, n.d.). These references enable SCA tools to identify potential security threats and license incompatibilities. Studies have shown that modern SCA tools differ significantly in their licensing models (commercial vs. open-source), supported programming languages, and the extent of their reporting capabilities (Ning et al., 2024).

Figure 1.1 presents a generalized SCA workflow, excluding the curation process that is incorporated in Figure 1.2, comprising several interlinked stages that enable the automated identification and risk evaluation of open-source components. The process begins with Dependency Scanning, where the source code is analyzed to detect both direct and transitive dependencies. This can be achieved using scanners such as ScanCode (n.d.), or FOSSology (n.d.), which focus on static license detection, or through integrated platforms like Snyk Open Source (n.d.), which combine scanning with continuous monitoring. These tools traverse file trees and extract metadata from headers, README files, comments, and package managers, aiming to uncover both licensing terms and known vulnerability signatures.

Following this, the Matching & Analysis Engine maps the detected components to known entries in vulnerability and license databases. These include the NVD, as maintained by NVD (n.d.), the SPDX standard (SPDX, n.d.), and various vendor-maintained security advisories. This step often applies heuristics or rule-based logic to resolve ambiguous cases, such as inconsistent versioning, unstructured license identifiers, or undocumented Common Vulnerabilities and Exposures (CVE) entries, ensuring accurate matching of both vulnerability and licensing metadata.

The resulting matches are then evaluated in the Risk Scoring & Policy Evaluation phase, where vulnerabilities are prioritized based on severity metrics like the Common Vulnerability Scoring System (CVSS), an industry-standard framework that assigns a numerical score (typically ranging from 0.0 to 10.0) and a qualitative severity label (e.g., Low, Medium, High, or Critical) based on factors such as exploitability, impact, and scope. Simultaneously, tools assess license risks, such as copyleft obligations or commercial incompatibilities, and verify whether component usage aligns with organizational policies.

Finally, the Reporting & Alerts stage communicates the findings, often integrating with Continuous Integration / Continuous Deployment (CI/CD) pipelines to notify developers, DevOps, and security stakeholders (Security, 2024). This structured workflow ensures that SCA tools not only detect and trace third-party code usage but also evaluate it comprehensively for both vulnerabilities and license compliance, enabling informed risk mitigation decisions.

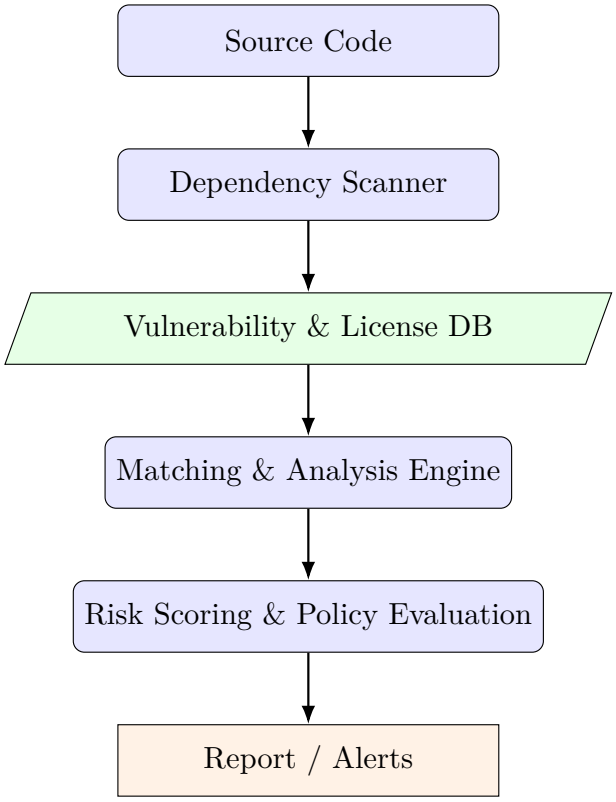


Figure 1.1: Typical SCA workflow adapted from Security (2024)

1.2 Understanding the Role of Curation in SCA

While the workflow shown in Figure 1.1 captures the technical pipeline, it omits a critical **curation** component. In the context of SCA, curation refers to the validation, correction, and enrichment of metadata used in detection and reporting. Poor curation can lead to false positives, missing vulnerabilities, or license misclassification, outcomes that ultimately undermine trust in the SCA tool's output.

> *As defined by Chen et al. (2020)*, in the context of software security, vulnerability curation involves maintaining and enhancing a database of open-source library vulnerabilities by utilizing information from various sources, including commit logs, bug reports, and mailing lists. More broadly, curation can be understood as the process of validating and refining data or metadata associated with components or vulnerabilities within a software system. It encompasses actions such as updating inaccurate license references, verifying scanner findings, and addressing false positives or negatives.

This definition is pivotal to the thesis and will serve as the foundation for evaluating SCA tool practices.

Figure 1.2 integrates this missing element by explicitly illustrating a feedback loop between the Vulnerability & License DB and a dedicated Curation Process. This placement reflects industry practices in which metadata such as license identifiers, CVE mappings, or component versions is actively maintained and refined outside of the scanning pipeline, then reintegrated into the core databases. For example, ClearlyDefined enables community submitted corrections to component metadata, which are reviewed and then fed back into its openly accessible database, improving downstream license analysis (ClearlyDefined Curation, n.d.).

While this diagram highlights pre-analysis curation that updates the central metadata repository, some tools also enable post-analysis curation or adjustments during the Matching & Analysis phase. For instance, Black Duck allows users to override vulnerability severity when the automated analysis produces incorrect or incomplete results (Synopsys, 2024). These corrections, such as resolving false positives or refining metadata matches, represent a form of curation, as they improve the accuracy of the scanner's output. According to Synopsys documentation, "the Override Severity dropdown menu is available to users with update role". It allows users to change the severity of all findings currently selected" (Synopsys, 2024). However, when overrides are applied to align with internal policy decisions rather than to correct inaccuracies, they reflect governance rather than metadata curation.

As described in Imtiaz et al. (2021), tools differ significantly in how they handle non-CVE vulnerabilities, overlapping component names, and external feeds. These differences highlight the importance of both pre and post analysis curation to reduce ambiguity and ensure meaningful results. Thus, the diagram reflects one key manifestation of curation while acknowledging that such practices may also take place during or after analysis, depending on the tool's design and intended use.

1.3 Problem Statement

SCA tools have become essential for identifying vulnerabilities and ensuring license compliance in software systems that rely on open source components. These tools operate by scanning codebases, identifying dependencies, and mapping them to databases that contain vulnerability and licensing information. However, the accuracy and reliability of SCA tools are fundamentally dependent on the quality of the metadata they consume.

Despite their critical role, SCA tools often lack transparent and standardized approaches to metadata curation, namely, the validation, correction, and updating

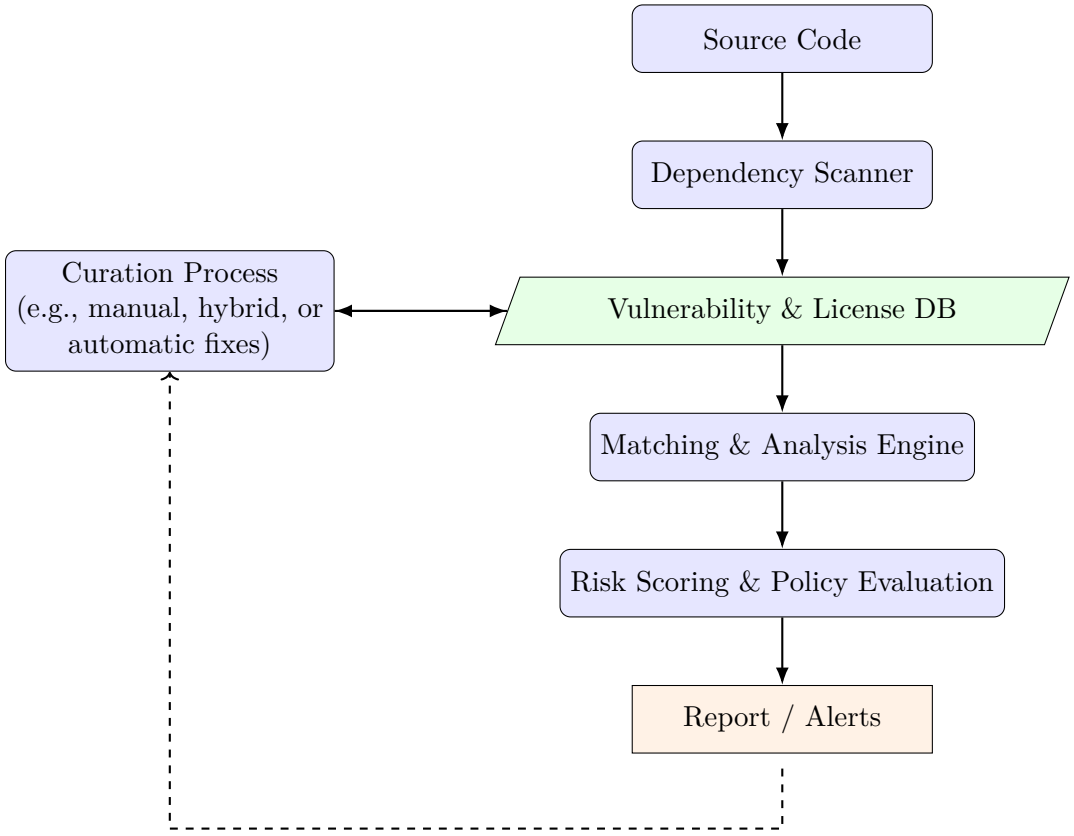


Figure 1.2: SCA workflow with curation implemented via a feedback loop

of license tags, component identifiers, and vulnerability mappings. Most tools rely on automated data retrieval from external sources with minimal documentation of how discrepancies are handled. This opacity leads to inconsistent outputs, including false positives, false negatives, and missed vulnerabilities, which hinder remediation efforts and reduce user trust. In the absence of robust curation mechanisms, SCA tools risk delivering unreliable results that compromise both security and legal compliance.

Furthermore, SCA tools differ substantially in their primary focus (e.g., vulnerability detection vs. license compliance), integration of data sources, update frequency, and support for manual, automatic, or hybrid metadata refinement. This variation complicates efforts to compare tools or assess their reliability, particularly when the underlying curation mechanisms are not well described in either scientific or vendor literature.

Existing research on SCA tools has largely emphasized detection capabilities and coverage, with limited attention given to metadata curation practices. As a result, the field lacks a structured understanding of how tools manage metadata integrity, how these practices vary, and what methods from other domains could

be adapted to improve curation in SCA contexts.

This thesis addresses this gap by systematically analyzing how current SCA tools implement metadata curation, examining both explicit documentation and inferred practices based on tool behavior and available literature. The study also explores relevant curation models from other domains to identify transferable strategies that can enhance metadata quality, consistency, and transparency in SCA tools.

1.4 Research Objectives

The thesis sets out to achieve the following objectives:

- Clearly define the concept of "curation" specifically within the context of SCA tools.
- Systematically review and categorize existing SCA tools according to their primary focus areas (e.g., vulnerability detection, or license compliance).
- Identify and critically analyze the curation practices described in both scientific and gray literature regarding SCA tools.
- Discuss curation practices from different fields and assess their applicability to improving metadata management in SCA tools.

Based on these objectives, the research questions formulated in Chapter 2 are directly derived to ensure a structured and goal-oriented investigation of curation practices within SCA tools.

The thesis structure is organized as follows: Chapter 2 reviews related literature; Chapter 3 details the research methodology employed, following the SLR guidelines proposed by Kitchenham (2004); Chapter 4 presents the research results, highlighting the categorization and curation practices of SCA tools; Chapter 5 discusses these findings, their implications, and study limitations; finally, Chapter 6 summarizes the thesis and proposes avenues for future research.

2 Related Work

2.1 Overview

This chapter critically reviews academic literature on SCA tools, with particular focus on how these tools have been evaluated, categorized, and discussed in relation to metadata curation. While existing studies offer valuable insights into detection performance and coverage, few explicitly examine how SCA tools curate the metadata they rely on, through processes such as validation, correction, and updating of vulnerability or license information.

The following section highlights this gap by analyzing selected studies that assess tool outputs or propose curation techniques, yet rarely trace discrepancies back to internal metadata workflows. This synthesis establishes the motivation for the present thesis: to investigate how metadata curation practices are implemented or omitted in both academic and commercial SCA tools, and to promote greater transparency and reliability in open-source security management.

2.2 How SCA Tools Have Been Evaluated: and What's Missing

Several studies have critically evaluated SCA tools, particularly their performance in vulnerability detection and license incompatibility identification, but have rarely examined the underlying curation processes that ensure metadata reliability. To structure the review, each of the following studies is assessed based on its evaluation of SCA tool performance, its treatment of metadata curation practices, and whether it introduces broader frameworks for categorization or evaluation.

Sharma et al. (2024) conducted a qualitative evaluation of five widely used SCA tools, Snyk Open Source, OWASP Dependency-Check (ODC), Open Source Vulnerabilities (OSV) Scanner, GitHub Dependabot, and Red Hat CodeReady Dependency Analytics, highlighting substantial discrepancies in vulnerability detec-

tion, transitive dependency coverage, and severity scoring. Their study focuses on practical feature comparisons, such as dependency resolution, vulnerability mapping, and Software Development Lifecycle (SDLC) integration, thereby contributing valuable insights into the operational differences among tools. Additionally, Sharma et al. (2024) use empirical clustering based on tool outputs to group SCA tools with similar detection behaviors, implicitly offering a categorization based on shared database usage and analysis strategies. However, the study does not examine metadata curation processes such as validation, correction, or enrichment of vulnerability records. Nor does it introduce theoretical frameworks for systematically evaluating or classifying SCA tools beyond the data-driven clustering. While the paper surfaces metadata quality inconsistencies, it does not investigate the internal workflows responsible for these variations. Building on these observations, the present thesis explicitly investigates metadata curation methodologies, addressing a critical but underexplored dimension of SCA tool reliability.

Imtiaz et al. (2021) conducted a comparative evaluation of nine industrial SCA tools, by analyzing their vulnerability reports on the OpenMRS project. Their study systematically highlights discrepancies in the detection of vulnerable dependencies, attributing these inconsistencies to differences in dependency detection mechanisms, coverage gaps, and the choice of vulnerability databases. While this evaluation provides valuable insights into the performance and limitations of SCA tools, it remains focused on empirical outcome comparison without introducing a formal theoretical framework or categorization model for software evaluation. Furthermore, the study does not explicitly investigate metadata curation methodologies; there is no analysis of how SCA tools validate, correct, or update vulnerability metadata internally. Although their findings implicitly underscore the significance of metadata quality for reliable vulnerability reporting, the curation processes behind metadata maintenance are not explored. Building upon these limitations, this thesis explicitly examines metadata curation strategies to fill this critical gap in the evaluation of SCA tools.

Bottner et al. (2023) conducted an empirical evaluation of two open-source SCA tools, Eclipse Steady and ODC, focusing on their effectiveness in detecting vulnerabilities in Java projects. Their study emphasizes the conceptual distinction between code-centric (Eclipse Steady) and metadata-centric ODC vulnerability detection approaches, and reveals that both methods suffer from substantial rates of false positives and false negatives. Particularly, the analysis underscores that detection quality is strongly influenced by the completeness and correctness of the underlying vulnerability databases NVD for ODC and Project Kaybee for Eclipse Steady). However, while Bottner et al. (2023) recognize the limitations in vulnerability database quality, such as missing entries, outdated vulnerability data, and inconsistencies, they do not systematically investigate or document how these databases are curated, validated, corrected, or updated by the tools.

Nor do they introduce a broader theoretical framework for categorizing SCA tools beyond the practical distinction between code-centric and metadata-centric approaches. Thus, although their work implicitly surfaces metadata reliability challenges, it stops short of examining the internal curation workflows critical for understanding and improving SCA tool accuracy. Building on these insights, this thesis explicitly investigates metadata curation methodologies, filling a significant gap in the evaluation of SCA tools.

Chen et al. (2020) introduce a hybrid vulnerability metadata curation pipeline that combines manual labeling with machine learning to support large-scale classification of vulnerability-related artifacts. Their system integrates weak labeling, feature extraction (including commit metadata and developer information), and self-training to automate validation and enrichment of vulnerability data sourced from bug trackers, mailing lists, reserved CVE entries, and version control systems. The approach aligns with this thesis's operational definition of metadata curation by supporting validation (via model-driven classification), correction (through iterative retraining), and enrichment (via contextual metadata integration). Notably, the method exemplifies a hybrid curation model, beginning with expert-verified data and extending via automated predictions.

However, while methodologically relevant, the paper does not apply this curation process within any specific SCA tool, nor does it evaluate how existing tools validate or maintain metadata. As such, it falls short of the full scope of this thesis, which focuses explicitly on understanding how metadata curation is operationalized in SCA tools. Nevertheless, the study offers a valuable technical foundation for scalable metadata curation and highlights the importance of automated support in improving metadata reliability, an issue central to this thesis.

2.3 Summary and Research Gap

The literature reviewed reveals that while substantial research has assessed the detection performance, dependency resolution, and practical integration of SCA tools, explicit investigations into metadata curation, understood here as the validation, correction, and enrichment of vulnerability and license metadata, remain rare. Studies such as Sharma et al. (2024), Imtiaz et al. (2021), and Bottner et al. (2023) primarily evaluate discrepancies and limitations in tool outputs (e.g., false positives, missing transitive dependencies, inconsistent severity scoring), yet stop short of analyzing the internal curation workflows that might explain or resolve such issues. As such, the root causes of these inconsistencies, such as outdated entries, misaligned vulnerability mappings, or uncurated license references, are acknowledged but not systematically explored.

Although Chen et al. (2020) introduce a generalizable machine learning based

methodology for automating vulnerability metadata classification, their approach is not applied to or evaluated within any specific SCA tool. While this contribution highlights the feasibility of scalable, hybrid curation mechanisms, it remains disconnected from the evaluation of how curation is operationalized in concrete SCA workflows. These studies, while advancing methodological innovation, fall short of addressing how curation practices are embedded in practice within academic or commercial SCA tools.

Moreover, existing categorization efforts, such as the clustering-based analysis in Sharma et al. (2024) or the metadata-centric vs. code-centric distinction in Bottner et al. (2023), are primarily empirical and outcome-driven. They lack formal frameworks for evaluating the extent and quality of metadata curation across tools. As a result, the critical infrastructure supporting trust in vulnerability and license data remains largely implicit and underexamined.

In this context, metadata curation within SCA tools functions much like the editorial mechanisms of Wikipedia: largely invisible, yet crucial to the credibility of the final output. This thesis addresses the resulting gap by systematically analyzing how metadata curation practices, manual, automated, or hybrid, are implemented, or omitted, across both academic prototypes and industry-grade SCA tools. In doing so, this thesis surfaces the often-overlooked curation processes embedded within SCA tools, contributing to more transparent, accurate, and trustworthy practices in OSS risk management.

3 Research design

The research design employed in this study is grounded in the SLR, conducted in accordance with the guidelines proposed by Kitchenham (2004) for performing systematic reviews in software engineering. This methodology enables a structured, rigorous, and replicable approach to identifying, selecting, and analyzing primary studies related to SCA tools, with a specific emphasis on curation practices for managing vulnerabilities and ensuring license compliance.

Figure 3.1 illustrates the three primary phases of the SLR process, Planning the Review, Conducting the Review, and Reporting the Review, as defined by Kitchenham (2004). These phases provide the conceptual foundation for the SLR design adopted in this thesis.

The subsequent sections of this chapter detail the implementation of these phases, including the formulation of research questions, the development of inclusion and exclusion criteria, and the execution of the literature search, study selection, and data extraction procedures.

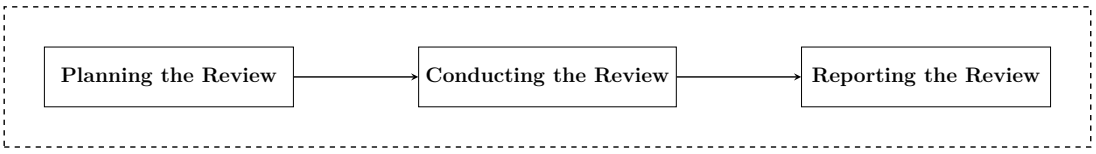


Figure 3.1: Primary Phases of a SLR based on Kitchenham (2004)

3.1 Planning the Review

Planning the review is the first and most important step of the SLR. It creates a strong base for the rest of the study. As Kitchenham (2004) points out, good planning helps reduce bias, makes the review easier to repeat, and leads to more useful and trustworthy results. In this study, the planning phase included three key steps: Defining the research questions, setting clear rules for which papers to include or leave out, and creating a strong search plan. These steps are shown in Figure 3.2.

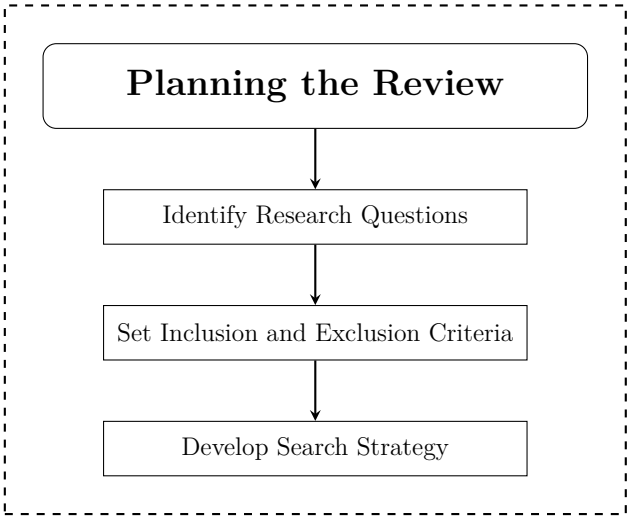


Figure 3.2: Planning phase of SLR based on Kitchenham (2004)

3.1.1 Identifying Research Questions

The research questions in this study aim to explore the currently underexplored practices surrounding how vulnerability and license data are curated in SCA tools. They are designed to give insights into both current practices and possible lessons from other fields. The following questions guide this research:

- **RQ1:** What are the current practices for curation of automatically retrieved data during SCA, such as copyright or licensing information?
- **RQ2:** What curation practices does the scientific and gray literature describe?
- **RQ3:** How have existing SCA tools been evaluated?
- **RQ4:** What are current curation practices in other fields that might be applicable to SCA tools?

These questions shape the literature review's structure and guide the subsequent analysis of tools and methodologies.

3.1.2 Inclusion & Exclusion Criteria

To ensure a systematic and unbiased selection of studies, clear inclusion and exclusion criteria were defined. These criteria guided both the initial title and abstract screening as well as the subsequent full-text eligibility assessment, with a specific focus on curation practices in SCA tools.

Inclusion Criteria

A study was included if it met at least two of the following conditions:

1. It introduced, described, or evaluated a standalone SCA tool, a related component (e.g., a license or vulnerability scanner), or a supporting framework designed to facilitate or extend SCA capabilities.
2. It explicitly or implicitly addressed curation practices in the context of SCA tools.
3. It discussed the data sources employed by SCA tools for tasks such as vulnerability detection or license compliance (e.g., NVD, SPDX, proprietary databases), or examined the frequency and mechanisms by which these sources are updated.
4. It originated from reputable sources of gray literature related to SCA, including vendor documentation, industry white papers, or platform reports listed in Table 3.2.

Exclusion Criteria

The following types of literature were excluded from the review:

1. Studies focused exclusively on detection algorithms, dependency analysis, or SCA tool functionality without explicitly or implicitly addressing metadata-related challenges, such as data source transparency, update frequency, or curation practices.
2. Grey literature lacking transparency, credibility, or technical relevance, such as unverified blog posts, marketing brochures, or opinion-based commentary without supporting evidence.
3. Publications not written in English.
4. Sources for which the full text was inaccessible after multiple retrieval attempts (e.g., due to persistent paywalls or broken access links).

Applying these exclusion criteria ensured that the final selection of studies remained methodologically rigorous and directly aligned with the research questions and objectives of this thesis.

3.1.3 Develop Search Strategy

Following the guidelines from Kitchenham (2004) for conducting systematic reviews in software engineering, a clear and step-by-step search strategy was developed to find literature about SCA tools and how metadata is curated in those

tools. Because the word “curation” is not often directly used in papers about SCA tools, a layered search method was used.

To ensure that the search specifically retrieved studies focusing on SCA tools rather than broader discussions of OSS practices, the final Boolean query incorporated the term ”Tool” into the metadata-related keywords. This adjustment was necessary to maintain alignment with the study’s focus on evaluating and analyzing SCA tools and to ensure the retrieval of key technical papers directly relevant to the research questions.

Final Query

The final search query was applied to the title field to enhance precision, and is formulated as follows:

> (”Software Composition Analysis” OR ”SCA tool” OR ”Open Source Software” OR ”open-source components” OR ”OSS”) AND
> (”curation” OR ”correction” OR ”updat*” OR ”vulnerab*” OR ”licens*” OR ”quality control” OR ”software quality” OR ”improvement” OR ”clearance” OR ”tool” OR ”scanner”)

Table 3.1: Search Query Refinement Process

Query Adjustment	**Field Searched Results**	
	All Fields	**Title Field**
Initial Query: `"Software Composition Analysis"` OR `"SCA tool"`	199	14
Above Query OR Open Source Software Terms: `"Open Source Software"` OR `"open-source components"` OR `"OSS"`	289,634	6,190
Above Query AND Curation-Related Terms: `"curation"` OR `"correction"` OR `Updat*`	54,261	31
Above Query OR Vulnerability/License Terms: `"vulnerab*"` OR `"licens*"`	73,920	166
Above Query OR Quality Improvement Terms: `"improvement"` OR `"quality control"` OR `"software quality"`	103,009	202
Above Query OR Final Terms: `"clearance"` OR `"tool"` OR `"scanner"`	164,519	379

This query was carefully designed to find as many relevant papers as possible (recall) while avoiding too many unrelated ones (precision). It includes a wide

range of keywords but limits the results to literature that mainly focuses on SCA tools. The search strategy went through several stages of refinement to improve both recall and precision. This iterative process is visualized in Figure 3.3, while Table 3.1 summarizes each query modification and its outcome. To capture multiple inflections of keywords and enhance recall without significantly reducing precision, wildcard symbols (e.g., vulnerab*, updat*) were incorporated into the final search query.

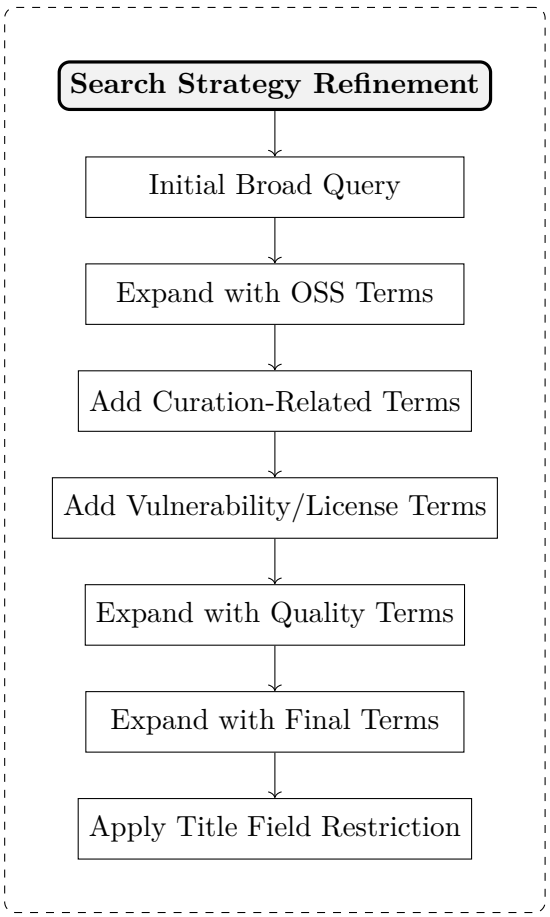


Figure 3.3: Refinement Steps in Search Strategy

Grey Literature Integration

Due to the limited availability of peer-reviewed publications explicitly addressing metadata curation practices in SCA tools, this study incorporates selected high-quality grey literature sources to complement the academic dataset. These sources were chosen based on their credibility, transparency, and direct relevance to the research questions, particularly in relation to how metadata is validated, corrected, and enriched.

By integrating these sources, the study captures a more comprehensive and nuanced understanding of current metadata curation practices across a range of tool types, including open-source, proprietary, and community-driven initiatives. Additionally, it draws on curation approaches from adjacent domains to provide transferable insights. Notably, *NeMo Curator* by NVIDIA (n.d.), a framework for managing and refining training data pipelines, was included to illustrate how metadata curation is handled in other technical fields.

Table 3.2: Grey Literature Sources Included in the Study

Grey Literature Source
Snyk Open Source
Black Duck by Synopsys
Sonatype Nexus Lifecycle
JFrog Xray
FOSSology
ClearlyDefined
NVIDIA NeMo Curator
Open Source Initiative

This methodical integration of grey literature ensured that both academic publications and grey literature sources were appropriately aligned with the objectives of this thesis. It also provided a practical foundation for understanding how metadata curation is implemented in SCA tools and highlighted relevant models and strategies for evaluation.

3.2 Conducting the Review

The conducting phase of this SLR, illustrated in Figure 3.4, represents the practical execution of the structured plan outlined during the planning stage (Section 3.1). Following the guidelines by Kitchenham (2004), this phase entailed a sequential process involving the identification of relevant literature, quality assessment, the application of inclusion and exclusion criteria, and systematic data extraction and synthesis.

3.2.1 Literature Identification

The search query formulated during the planning phase (Section 3.1) was executed in the Scopus[1] database. A multi-stage review process followed, consisting of title screening, abstract evaluation, and full-text analysis. The use of the

[1] https://www.scopus.com/

`ALLINTITLE` operator improved search precision by focusing on literature where SCA tool or metadata-related practices were explicitly mentioned in titles. This approach was necessary because initial exploratory searches without title restrictions returned tens of thousands of articles, making manual screening impractical and introducing a high risk of selection bias. By restricting the query to titles, the number of candidate studies became manageable, allowing for a systematic and replicable screening process.

However, to mitigate the known limitation of the operator, namely, its tendency to overlook papers that discuss curation only within the body text, the process was supplemented by expanded keyword variations.

Given the scarcity of direct mentions of curation, additional high-quality grey literature was considered. The criteria for including such literature is discussed in the following subsection.

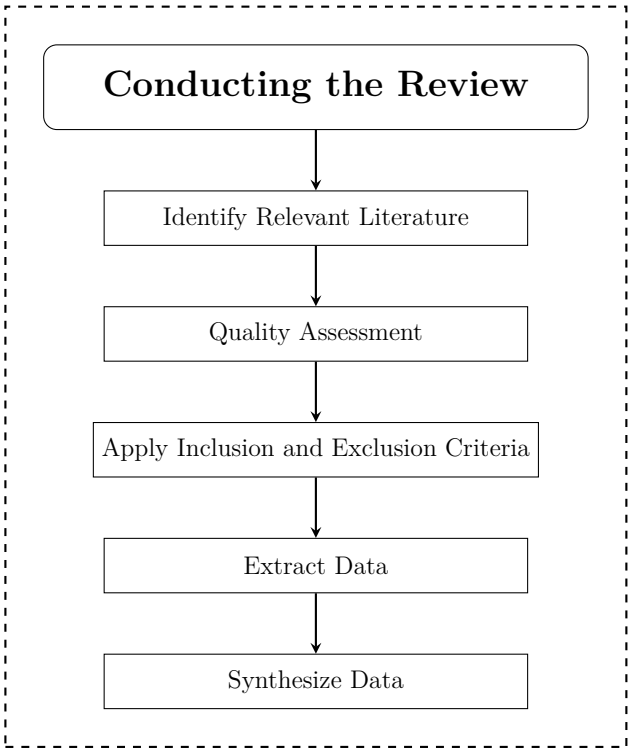


Figure 3.4: Conducting Phase of the SLR based on Kitchenham (2004)

3.2.2 Quality Assessment

While a comprehensive quality assessment framework was considered, the scarcity of literature explicitly addressing metadata curation in SCA tools limited the extent of formal evaluation. Therefore, a single pragmatic criterion was applied during screening: studies were included if they provided either a conceptual or

practical discussion relevant to metadata validation, correction, or update mechanisms in SCA tools, regardless of whether the term "curation" was explicitly used.

Because explicit references to "curation" or "correction" were largely absent from most reviewed sources, interpretive judgments were necessary. In such cases, inclusion decisions were based on whether the study described activities implying metadata management, such as license validation, mitigation of false positives, or updating vulnerability databases. This interpretive flexibility ensured that relevant contributions were not overlooked due to terminological inconsistencies.

3.2.3 Application of Inclusion and Exclusion Criteria

Following quality assessment, the application of inclusion and exclusion criteria served as a second level of filtering to ensure that all retained literature meaningfully contributed to the research objectives. While these criteria were initially defined during the planning phase (Section 3.1.2), their practical implementation required iterative interpretation due to the limited number of publications explicitly discussing metadata curation in SCA tools.

Papers were included if they addressed issues related to metadata accuracy, validation, correction, or maintenance within SCA contexts, even when not directly framed using the term "curation." For example, studies focusing on topics such as license misclassification, false positives in vulnerability detection, or update mechanisms for vulnerability databases were considered relevant. Additionally, literature describing specific tool components, such as Ninka's automated license rule management system described in Higashi et al. (2023), was also retained.

Conversely, publications that focused solely on detection algorithms or dependency resolution, without engaging with metadata quality improvement practices, were excluded. Grey literature sources that lacked reputability, technical rigor, or transparency (e.g., informal blogs or marketing brochures) were similarly excluded.

This pragmatic and flexible application of inclusion and exclusion criteria ensured that the final corpus of studies encompassed both direct evaluations of SCA tools and conceptual contributions to the advancement of metadata curation practices within the SCA domain.

3.2.4 Data Extraction and Synthesis

To ensure replicability and traceability, a data extraction form was developed based on Kitchenham (2004)'s SLR methodology. This form captured both tool-specific and methodology-specific attributes relevant to curation within SCA tools.

The design of the data extraction form was informed by the core objectives of this study and aligned with Kitchenham's systematic review principles. The form was structured to capture multiple dimensions of each selected source, with particular attention to the primary focus of the tool under discussion, such as whether it prioritized vulnerability detection, or license compliance. It also recorded the type of curation methodology employed, distinguishing between manual processes, automated approaches, or hybrid models that combine both. Additionally, the extraction form included fields to capture update frequency and the provenance of metadata sources, ranging from standardized repositories such as the NVD, as described by NVD (n.d.), and SPDX (n.d.), to community-driven platforms such as GitHub.

Finally, it tracked whether metadata curation was addressed explicitly in the literature or had to be inferred based on the description of tool functionality. As the review progressed, these fields were refined iteratively to reflect recurring themes and emerging insights, ensuring flexibility in capturing both peer-reviewed and grey literature. An initial pilot extraction on a subset of papers was conducted to validate the comprehensiveness of the fields, after which minor adjustments were made. The finalized fields are summarized in Table 3.3, while the full coding framework, including all categories and code definitions, is provided in Appendix C to ensure traceability and reproducibility.

Table 3.3: Data Fields for SCA Tools Analysis

Field	**Description**
Tool/Component Name	Name of the SCA tool/component.
Curation Methodology	Automated, manual, or hybrid.
Data Sources	Databases used (e.g., NVD, SPDX, or propriety).
Update Frequency	How often metadata is updated.
Focus Area	Whether the tool focuses on vulnerabilities, or licenses.
Evaluation Methodology	How the tool was assessed in the literature.
Curation Mentioned?	Whether curation is explicitly discussed.

Use of MAXQDA for Thematic Coding

MAXQDA[2] was employed to facilitate structured data extraction and thematic synthesis. Compared to tools such as QDAcity[3], *MAXQDA* offered more advanced capabilities for mixed-method analysis, visualization, and offline data

[2]https://www.maxqda.com/
[3]https://qdacity.com/

management. It enabled precise code application, annotation of relevant excerpts, and the creation of traceability links between literature content and extracted data.

Challenges and Adaptations in Data Extraction

During data extraction, several methodological challenges emerged, necessitating adaptive strategies.

- **Terminological inconsistency:** Most papers did not use the term "curation." To address this, the extraction vocabulary was expanded to include related terms such as "metadata correction," and "updation."
- **Lack of direct evaluation metrics for curation:** Most studies emphasized tool performance metrics such as vulnerability detection rates or dependency resolution accuracy, without directly assessing metadata curation quality. To address this gap, proxy indicators, specifically update frequency and the degree of community involvement in metadata maintenance, were systematically extracted. These proxies were selected because they provide indirect yet meaningful insights into the timeliness, reliability, and collaborative validation practices associated with curation activities in SCA tools.
- **Partial tool coverage:** Some studies focused only on components (e.g., license scanning modules) rather than complete SCA tools. For example, Higashi et al. (2023) describe an automation technique for maintaining license detection rules in Ninka, a module widely used in SCA workflows. Such studies were included due to their conceptual contribution to curation practices in SCA tools.

Thematic Synthesis and Categorization

The data collected was analyzed through thematic synthesis to identify recurring themes, knowledge gaps, and methodological trade-offs in metadata curation practices within SCA tools. Several key patterns emerged:

- **Primary Focus of SCA Tools:** Tools were categorized based on their main focus area, vulnerability detection, or license compliance, since focus areas often influenced the types and priorities of curation activities.
- **Levels of Automation in Curation:** Tools varied in the degree of automation applied to curation processes, ranging from purely manual validation to fully automated pipelines or hybrid approaches combining human and machine-based validation.
- **Transparency and Auditability of Metadata Updates:** Significant

differences were observed in the transparency of how metadata corrections and updates were documented and shared, especially between proprietary and open-source tools.

- **Proprietary vs. Open-Source Curation Models:** Commercial tools typically embedded curation practices within closed internal workflows, while open-source initiatives like ClearlyDefined promoted community-driven, transparent correction mechanisms.
- **Update Frequency and Data Source Provenance:** The extent to which metadata was regularly updated, and the reliability of sources such as the NVD, GHSA, or community-maintained repositories, were important indicators of curation quality.
- **Challenges in Identifying Curation Activities:** In many cases, curation practices were not explicitly labeled as such. Instead, evidence of validation, correction, or metadata enrichment had to be inferred from descriptions of tool features or update processes.
- **Evaluation Methods for Curation:** Few studies directly evaluated the effectiveness of curation activities. Proxy indicators such as update frequency, community involvement, or reduction of false positives were used to indirectly assess curation quality.

The objective of the synthesis was not merely to describe existing tools, but to critically analyze the methodological trade-offs shaping metadata curation practices within SCA tools. The detailed categorization outcomes and findings are presented in the Research Results chapter (Chapter 4).

3.3 Reporting the Review

The final phase of the SLR, illustrated in Figure 3.5, involves organizing and presenting the synthesized findings in a clear, coherent, and reproducible manner. As recommended by Kitchenham (2004), reporting is not merely a summary of included studies but a structured exposition that demonstrates how the literature was analyzed, how decisions were made throughout the process, and how the research questions were addressed.

3.3.1 Documenting Search and Selection Outcomes

The reporting phase begins with a transparent summary of how studies were selected. Although precise numeric filtering details are reserved for the Research Results chapter, this section clarifies that literature was retrieved through a Scopus search query (see Section 3.1.3), refined using the `ALLINTITLE` operator for precision. To mitigate the known limitations of this operator, such as excluding

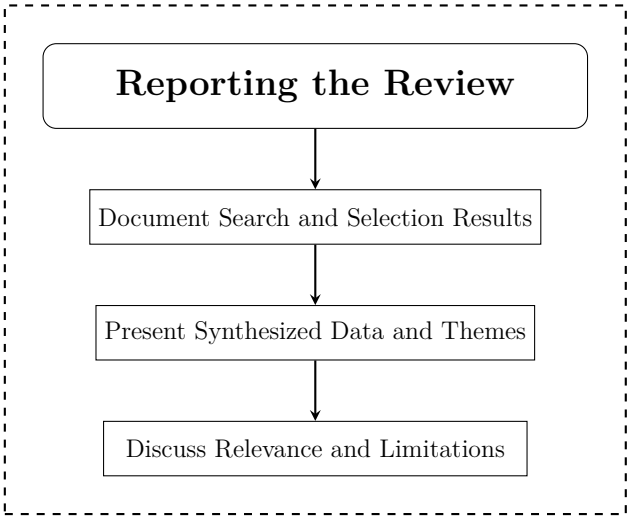


Figure 3.5: Reporting Phase of SLR based on Kitchenham (2004)

papers with relevant content not featured in the title, supplementary keyword searches, were employed.

Each document underwent a quality appraisal assessing its source credibility, methodological transparency, and relevance to metadata curation practices in SCA tools. Grey literature was carefully selected from trusted industry and community sources. These sources added valuable information and helped cover critical areas missing in academic research.

3.3.2 Presenting Synthesized Findings

The synthesized findings are structured thematically to align with the research questions outlined in the planning phase. As detailed in Section 3.2.4, data was systematically extracted using a structured form and coded using MAXQDA to support traceability and thematic analysis.

The classification of tools and approaches is organized along key analytical dimensions, including:

- Whether metadata validation, correction, and updating practices are performed manually, automatically, or through a hybrid approach.
- How frequently metadata is refreshed and whether updates rely on standardized databases, vendor-maintained sources, or community contributions.
- Whether the tool primarily targets license compliance, or vulnerability tracking.

To facilitate comparative analysis, these dimensions are presented through struc-

tured tables and figures in the Research Results chapter. This approach ensures a transparent, replicable synthesis of how different SCA tools address metadata curation challenges, including differences based on their focus area (license, or vulnerability), fulfilling the thesis objective of systematically categorizing and evaluating curation practices within the diverse landscape of SCA tools.

3.3.3 Relevance and Limitations

This review provides an in-depth understanding of how metadata validation, correction, and update processes are handled within SCA tools. The inclusion of conceptual contributions (e.g., frameworks not tied to a specific tool) and component-level studies (e.g., license scanners like Ninka) ensures that the study captures a broader picture of the curation landscape.

Nevertheless, certain limitations are acknowledged:

- The use of `ALLINTITLE` may have inadvertently excluded relevant studies that discuss metadata handling in the body but not in the title.
- The absence of standardized terminology (e.g., "curation" is often implied but not stated) made systematic filtering more complex and required thematic interpretation.
- Grey literature was necessary to supplement academic gaps; however, all sources were selected based on defined quality criteria to mitigate variability in rigor.

These limitations were mitigated by broadening inclusion criteria, and using MAXQDA to support transparent and traceable data coding. Overall, the reporting process ensures that the findings of this study are presented in a systematic, reproducible, and methodologically sound manner, aligned with the defined research objectives.

In summary, this chapter has detailed the structured methodology underpinning this study's investigation into metadata curation practices within SCA tools. By combining a carefully designed search strategy, systematic inclusion and exclusion processes, rigorous data extraction, and thematic synthesis, the approach ensures methodological rigor and replicability. The subsequent Research Results chapter builds directly on these foundations, presents the synthesized findings and interprets how metadata curation is or is not integrated into SCA tools. These insights help bridge gaps in current tool evaluation methodologies and support the development of more transparent and trustworthy software supply chain practices.

4 Research Results

This chapter presents the core findings of the SLR, executed in accordance with the methodological framework outlined in Chapter 3, based on the guidelines proposed by Kitchenham (2004). The purpose of this chapter is to synthesize and critically interpret how SCA tools address the problem of metadata curation, defined in this thesis as the processes of validating, correcting, and updating metadata related to licenses and vulnerabilities within OSS components.

The analysis proceeds in several stages. First, the reviewed literature is examined to identify and describe the SCA related tools and components. These tools are then categorized based on their primary focus: vulnerability detection, or license compliance. This categorization includes both standalone SCA tools, those providing full functionality for license or vulnerability analysis, and component-level tools, which support specific stages of the SCA pipeline (e.g., license rule engines, patch matchers) but do not operate as complete SCA tools. This distinction aligns with the inclusion criteria discussed in Section 3.1.2, which explicitly allowed for the inclusion of component-level studies when they provide meaningful insights into metadata curation.

In addition to tools and components, the review includes curated datasets and benchmarking frameworks that, while not SCA tools themselves, are highly relevant to understanding or evaluating curation practices. These include the manually curated vulnerability-fix dataset by Ponta et al. (2019), the Achilles benchmarking suite by Dann et al. (2022), and the automatically generated CVEfixes dataset by Bhandari et al. (2021). Each was included not as an operational SCA tool, but due to its value in illustrating how metadata curation can be implemented, validated, or supported in practical SCA contexts.

Second, each identified tool, component, or resource is evaluated in terms of how it supports metadata curation. Following the broader interpretation developed in Chapter 3 (Section 3.2.4), curation is not restricted to explicit use of the term, but encompasses any automated or manual activity that validates, corrects, or enriches metadata associated with license or vulnerability information. This includes actions such as resolving false positives, clarifying license classifications,

correcting scanner outputs, and making raw metadata more useful by analyzing its meaning or how it's used in code. For each included study, whether it describes a standalone SCA tool, a supporting component such as a scanner or classifier, or a curated resource like a dataset or benchmarking suite, this chapter evaluates how curation is performed, whether manually, automatically, or using a mix of both. It also considers how often the curated information is updated and whether the tool relies on open sources such as the NVD, as maintained by NVD (n.d.), the SPDX standard (SPDX, n.d.), or platforms like GitHub, or instead on private databases maintained by vendors.

Finally, in response to the limitations found in the academic literature, especially the limited availability of transparent and complete curation processes, this chapter expands its scope to include relevant information from high-quality grey literature. This includes both commercial and non-commercial tools, providing a broader understanding of real-world curation practices in SCA.

Several commercial tools, such as Snyk Open Source (n.d.), Black Duck (n.d.), Sonatype Nexus Lifecycle (n.d.), and JFrog Xray (n.d.), demonstrate how automated scanning, curated databases, and expert review can be combined to support accurate and scalable curation of metadata in production settings. In addition, this chapter also includes non-commercial tools that support open and transparent curation processes. FOSSology (n.d.), developed by the Linux Foundation, focuses on license compliance and allows users to manually validate and correct license findings through an interactive interface. ClearlyDefined (n.d.), a community-led project, encourages public contributions to improve the quality of open source metadata. It uses a structured process where contributors propose updates, which are then reviewed and merged using version-controlled workflows. These tools were included because they offer valuable insights into collaborative and reproducible approaches to metadata curation, which complement the closed practices of commercial platforms.

To complement these findings, the chapter also considers modular curation frameworks from other domains, including the QURATOR project by Rehm et al. (2020) and NeMo Curator by NVIDIA (n.d.). These highlight the role of semantically enriched and reproducible workflows in large-scale metadata management. By synthesizing lessons across domains, this chapter identifies key gaps and opportunities for advancing curation practices in SCA tools.

4.1 Search Results

Following the procedures defined in Sections 3.1.3 and 3.2 of the research design, the literature search was conducted using Scopus as the primary academic database. The search query was developed iteratively, combining specific keywords

related to SCA tools with broader terms relevant to metadata curation. To ensure comprehensive coverage, the search was supplemented by backward and forward snowballing, as well as by exploring additional keyword variations.

As part of the eligibility screening process, an initial set of 378 articles retrieved from Scopus was subjected to the inclusion and exclusion criteria outlined in Section 3.1.2. During this initial step, non-English publications were excluded, reducing the dataset to 369 articles. A title-based relevance screening was then performed, narrowing the pool to 138 articles. This was followed by an abstract-level assessment, which further reduced the selection to 58 articles. Finally, after conducting a full-text review, 25 articles were identified as fully relevant to the research questions and retained for detailed analysis, as illustrated in Figure 4.1.

In addition to database searches, backward and forward snowballing techniques were used to identify additional studies that could inform the broader context of the research. These supplementary efforts helped deepen the understanding of adjacent issues such as license and vulnerability incompatibilities, which are central to evaluating SCA tool curation practices. One article identified through this process was included for consideration in the final pool of analyzed papers. In contrast, the study by Chen et al. (2020), although highly relevant, was initially considered for inclusion but ultimately excluded from the final set of 25 articles, as it did not meet the inclusion criteria outlined in Section 3.1.2, specifically, it does not describe or evaluate a specific SCA tool or its component. Nonetheless, the work was incorporated into the related work chapter because it introduces a hybrid machine learning-based curation pipeline that aligns with the thesis's operational definition of metadata curation. While it does not focus on a particular SCA tool, its methodology offers valuable insights into scalable vulnerability metadata curation strategies that are broadly applicable across the SCA landscape.

The final selection comprised 25 articles, including 13 papers that introduce or evaluate standalone SCA tools, 9 papers that discuss supporting components enhancing SCA tool functionality, and 3 papers that present specialized resources relevant to the broader SCA workflow. A curated set of high-quality grey literature sources was also included to complement the academic findings (see Table 3.2).

Temporal Distribution of Included Papers

The temporal analysis of the included literature provides insight into the evolution of research interest in SCA tools. Figure 4.2 presents the distribution of articles by publication year. As shown, interest in this domain has accelerated significantly in recent years, with the highest number of relevant publications appearing in 2023 (7 papers), followed by 2024 (5 papers) and 2022 (4 papers). Only a few works

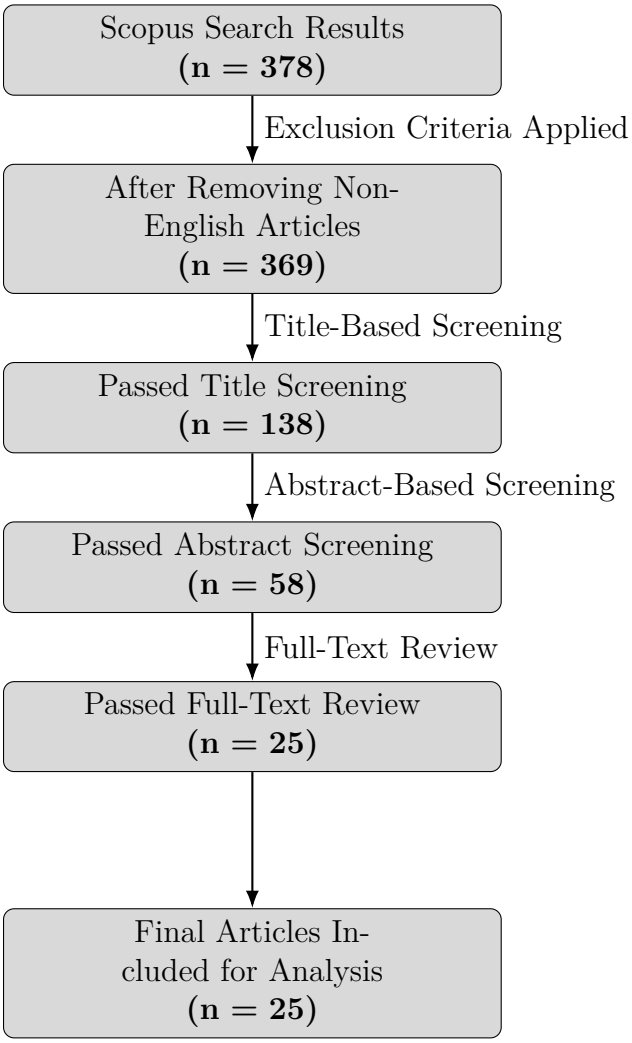


Figure 4.1: Flow diagram of article screening and selection process

were published before 2019, indicating that SCA tools gain more prominence only recently, likely due to the rising complexity and scale of OSS dependencies.

4.2 Tools Discussed in the Literature

Building on the research design outlined in Chapter 3, this section presents the SCA solutions identified through the systematic review. These include both standalone tools and supporting components relevant to metadata curation workflows.

To maintain consistency throughout this thesis, three terms are defined as follows. *Standalone SCA tools* refer to complete, end-to-end systems capable of performing license or vulnerability analysis independently. *Supporting compon-*

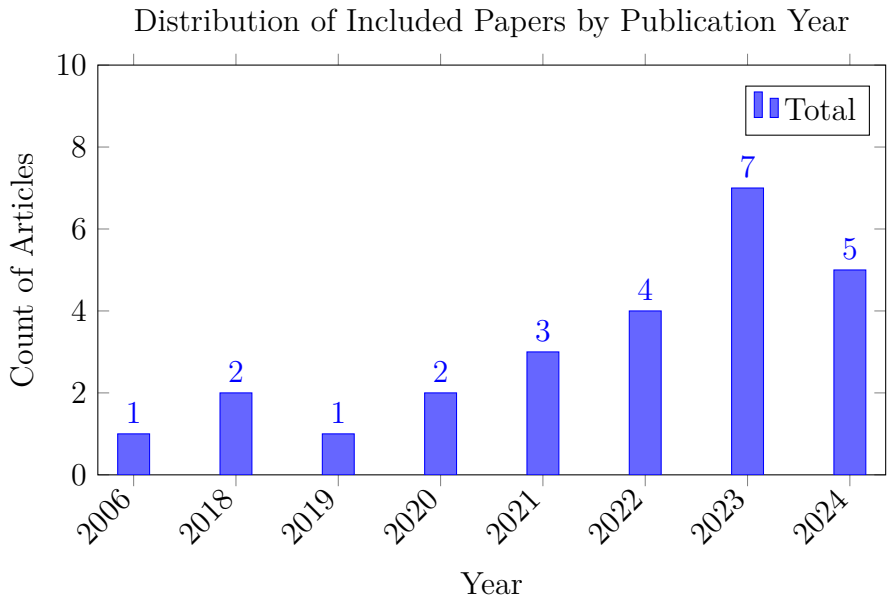


Figure 4.2: Distribution of Included Papers by Publication Year

ents denote task-specific modules, such as license scanners or patch matchers, that are often integrated into or used alongside larger tools. The term *SCA solution* is used in this thesis as an umbrella term encompassing both standalone tools and supporting components, as they all contribute to SCA and metadata curation.

4.2.1 Standalone SCA Tools

This subsection presents the standalone SCA tools identified in the reviewed literature. Thirteen papers introduced or evaluated standalone SCA tools that perform vulnerability detection or license compliance via automated techniques such as static code analysis, metadata matching, or dependency graph inspection.

Table 4.1 lists these tools and their associated references. Several studies conducted comparative evaluations. For instance, Imtiaz et al. (2021) assessed nine tools, including Snyk Open Source and ODC, focusing on their vulnerability reporting consistency.

It is important to note that some lightweight utilities are also grouped with more comprehensive SCA tools in the literature. For example, Ombredanne (2020) discusses GitHub Licensee alongside FOSSology and ScanCode in the context of license detection. However, GitHub Licensee[1] is best understood as a minimal tool that identifies a project's primary license by comparing its LICENSE file to

[1]https://github.com/licensee/licensee

known SPDX texts. Unlike standalone SCA tools such as FOSSology (n.d.) or ScanCode (n.d.), it does not inspect dependencies or analyze license metadata across an entire codebase.

Table 4.1: Standalone SCA tools and References

Tool Name(s)	Citation
LiDetector	Xu et al. (2023)
V1SCAN	Woo et al. (2023)
Holmes	Wu et al. (2024)
ODC, Dependabot, Grype, OSV Scanner, Snyk Open Source	Ivanova et al. (2024)
Eclipse Steady, ODC	Bottner et al. (2023)
ODC, Eclipse Steady, Dependabot Alerts, OSS Index, T1, T2	L. Zhao et al. (2023)
Black Duck, Cybellum, Scantist	Ning et al. (2024)
VODA	Kumar et al. (2024)
Vul4Java	Wang et al. (2024)
Vulas	Ponta et al. (2018)
ODC, Snyk Open Source, GitHub Dependabot, npmaudit, Eclipse Steady, Commercial A, Commercial B, Commercial C	Imtiaz et al. (2021)
FOSSology, ScanCode, GitHub licensee	Ombredanne (2020)
ODC, Eclipse Steady, C3	Dann et al. (2022)

4.2.2 Supporting Components Used in SCA Workflows

This subsection covers **9 additional papers** that present supporting components, or modular tools that assist core SCA tasks but are not complete tools themselves.

Since the term "curation" is rarely used explicitly in academic literature, inclusion of these works followed an interpretive approach (as described in Chapter 3). Studies were considered relevant if they addressed key curation activities such as license or vulnerability validation, metadata correction, dataset enrichment, or reducing false positives.

These supporting components fill essential gaps left by standalone tools, often offering refined analysis, patch-linking, or semantic validation capabilities. Their

Table 4.2: Supporting Components and Their Focus Areas

SCA Component Name	Focus Area	Citation
PatchScout	Vulnerability	Tan et al. (2021)
VERJava	Vulnerability	Sun et al. (2022)
DIKE	License	Cui et al. (2023)
Movery,	Vulnerability	Woo et al. (2022)
Ninka	License	Higashi et al. (2023)
findOSSLicense	License	Kapitsaki et al. (2022)
FOSSLT	License	J. Zhao et al. (2023)
osslist-maven-plugin (OMP)	License	Dyck et al. (2018)
ASLA	License	Tuunanen et al. (2006)

role in improving metadata quality justifies their inclusion despite their limited scope.

Table 4.2 summarizes these tools and their focus areas.

4.3 Tool Type Categorization

This section focuses on the unique SCA solutions identified in the reviewed literature, categorized by their primary functional focus: either vulnerability detection or license compliance.

As defined in Section 4.2, the term *SCA solutions* encompasses both standalone SCA tools and supporting components. Each solution was classified based on its dominant functionality as described in the corresponding study, regardless of whether it supports multiple functionalities. Accordingly, two focus areas are used throughout this section: vulnerability-focused and license-focused.

4.3.1 Vulnerability-Focused SCA tools

Vulnerability-focused SCA tools are primarily designed to detect known security vulnerabilities in OSS by linking declared or detected dependencies to entries in vulnerability databases such as the NVD, GHSA, or proprietary feeds. These tools aim to enhance software security by flagging components with known CVE and often provide actionable remediation guidance, such as upgrade recommendations or patch suggestions. For example, Snyk Open Source integrates both public and proprietary vulnerability intelligence to analyze project dependencies and generate remediation advice (Snyk Curation, n.d.). As shown by Ivanova et al.

(2024), Snyk Open Source and other metadata-reliant tools were evaluated specifically for their ability to detect vulnerabilities under manipulated conditions, highlighting both their capabilities and limitations in vulnerability detection.

4.3.2 License-Focused SCA Tools

License-focused SCA tools aim to ensure open-source license compliance by identifying and interpreting licensing terms embedded in source code or metadata. These tools typically detect license texts or identifiers and compare them against known reference templates or SPDX databases to assess compatibility, detect conflicts, and generate compliance reports, as demonstrated by Kapitsaki et al. (2022) and Cui et al. (2023).

In practice, license-focused tools may occasionally misidentify similar permissive licenses. For instance, scanners can confuse the Berkeley Software Distribution 3-Clause License (BSD-3-Clause) License with the MIT License due to their structural similarities. Correcting such misidentifications forms an important part of license metadata curation, which ensures that compliance reports reflect the project's actual legal obligations.

Beyond misidentification, the literature has also emphasized the risk of license conflicts arising from incompatible licensing terms. For example, Cui et al. (2023) demonstrate that combining code licensed under the GNU General Public License (GPL) version 3.0 or later with code licensed under the permissive MIT License can lead to a conflict. This is because the GPL requires derivative works to be distributed under the same license, while MIT imposes minimal redistribution restrictions. Tools like DIKE, presented by Cui et al. (2023), detect such conflicts by matching license conditions against a curated knowledge base of license compatibility rules.

Such tools play a critical role in identifying legal risks during software integration, especially in complex projects using multiple open-source dependencies.

Observed Trends

As summarized in Table 4.3, the distribution of SCA solutions examined in the reviewed literature reveals a pronounced emphasis on vulnerability analysis. Of the 29 unique SCA solutions, comprising both standalone tools and supporting components, 19 are categorized as primarily vulnerability-focused and 10 as license-focused. This distribution is visually represented in Figure 4.3, which shows two separate bars reflecting the relative prevalence of each category.

Table 4.3: Primary Focus Area of Analyzed SCA Solutions

SCA Solution	License-Focused	Vulnerability-Focused
LiDetector	✓	
DIKE	✓	
ScanCode	✓	
FOSSology	✓	
V1SCAN		✓
ODC		✓
GitHub Dependabot		✓
Grype		✓
OSV Scanner		✓
Snyk Open Source		✓
VULAS		✓
Black Duck		✓
Cybellum		✓
Scantist		✓
VODA		✓
Vul4Java		✓
OSS Index		✓
PatchScout		✓
VERJava		✓
Movery		✓
Ninka	✓	
Holmes		✓
findOSSLicense	✓	
FOSSLT	✓	
OMP	✓	
ASLA	✓	
Eclipse Steady		✓
GitHub Licensee	✓	
npm audit		✓

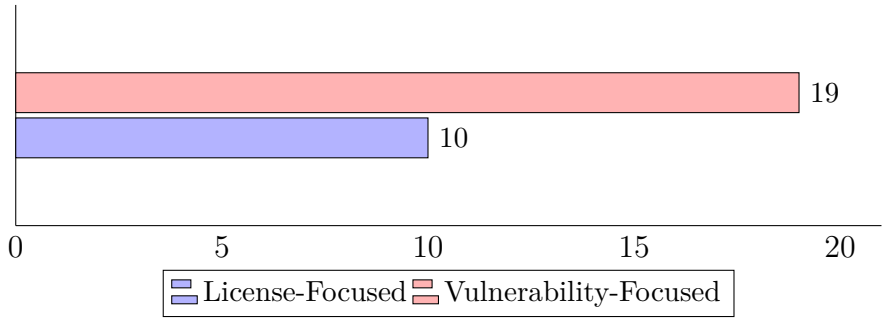
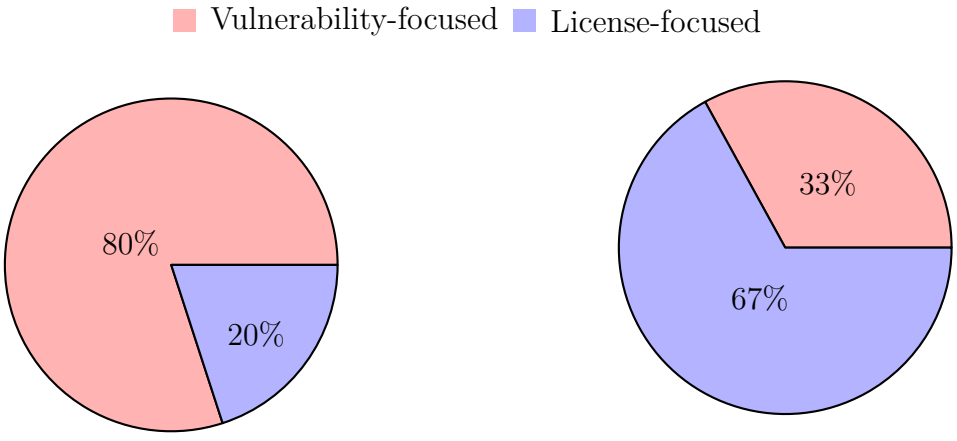


Figure 4.3: Distribution of SCA Solutions by Primary Focus

To better understand the underlying distribution, each standalone tool and supporting component was examined individually to identify its primary focus area. This detailed breakdown helps uncover potential patterns in how functionality is divided across different types of SCA solutions. As shown in Table 4.3, the 20 standalone SCA tools, including GitHub Licensee[1] display a clear emphasis on vulnerability detection: 16 are classified as vulnerability-focused, while only 4 focus on license compliance. In contrast, among the 9 supporting components, 6 focus on license-related tasks and only 3 address vulnerabilities. This distinction suggests that while standalone tools prioritize security concerns, license-related functionality often appears in more modular, lightweight components. The reversed trend among components is further visualized in Figure 4.4. A full listing of supporting components and their focus is provided in Table 4.2.

This observed difference in focus distribution across standalone tools and supporting components is reported as part of the empirical results and interpreted further in the Chapter 5.



(a) Standalone SCA tools **(b)** Supporting Components

Figure 4.4: Distribution of primary focus across analyzed SCA solutions

4.4 Curation Practices in SCA Tools and Components

Building on the tool categorization introduced in Section 4.3 and the screening criteria outlined in Chapter 3, this section presents a structured analysis of how metadata curation practices are addressed across the reviewed academic literature. In the context of this thesis, curation refers to the processes of validating, correcting, and enriching metadata associated with software licenses and vulnerabilities. Within SCA tools, this includes verifying the accuracy of scanner outputs, resolving misclassifications and false positives, and augmenting incomplete or ambiguous metadata.

To facilitate comparison, curation methodologies in the literature are categorized into three types:

- **Manual curation**: Expert-driven validation and correction of metadata, often requiring domain knowledge of codebases, licensing, or vulnerability semantics.
- **Automated curation**: Algorithmic refinement of metadata through rule-based heuristics or learning-based classifiers, performed without human intervention.
- **Hybrid curation**: A combination of automated techniques followed by human oversight, typically applied in scenarios involving ambiguous or low-confidence data.

While few studies explicitly employ the term "curation", a close examination of tool workflows reveals a range of practices that reflect curatorial intent, whether manual, automated, or hybrid.

Scope of Curation Analysis

Before presenting the identified curation practices across different SCA solutions, it is important to clarify the scope and selection criteria applied. Special emphasis was placed on papers that focus on a single tool or component, rather than on comparative studies evaluating multiple tools. This decision stems from the observation that comparative studies, such as Imtiaz et al. (2021), primarily evaluate tools based on output-level metrics like vulnerability detection accuracy, database coverage, and dependency reporting, rather than providing insight into internal curation methodologies such as validation, correction, or enrichment.

When multiple tools are compared, the treatment of curation practices is often superficial, making it challenging to isolate and extract detailed, tool-specific curation strategies. Nevertheless, such comparative studies remain valuable for

identifying broader aspects relevant to this thesis, such as the data sources employed by SCA tools and the frequency of database updates. These aspects have been referenced in appropriate contexts throughout the results.

Additionally, this study includes papers that do not introduce SCA solutions according to the criteria outlined in Section 4.2, but still offer valuable insights into metadata curation practices. These include discussions of Dann et al. (2022)'s *Achilles*, a test suite derived from manually validated vulnerability data, and curated datasets such as Bhandari et al. (2021)'s *CVEfixes* and Ponta et al. (2019)'s manually collected fix-commit dataset. These works illustrate academic approaches to validating and enriching vulnerability metadata. They are especially relevant for addressing the research questions concerning current curation practices in SCA, and they will be discussed in more detail in subsequent sections.

Furthermore, commercial tools such as Snyk Open Source, and Black Duck were deliberately excluded from the academic literature review, as their curation practices are more comprehensively documented in grey literature (e.g., vendor blogs, and documentation). These sources are analyzed in a separate section that follows.

The next sections thus focus exclusively on tools and components where curation practices could be meaningfully identified, ensuring that the analysis remains rigorous and aligned with the research objectives. The discussion is structured according to the operational definitions introduced earlier, beginning with manual curation practices observed in the academic literature.

4.4.1 Manual Curation Practices across the Literature

The following section examines cases where SCA tools rely primarily on human intervention to validate, correct, or enrich metadata.

Manual Curation of Vulnerability Metadata in VERJava

Sun et al. (2022) introduce VERJava, a lightweight tool for identifying vulnerable versions of Java OSS projects affected by known CVE entries. To construct a reliable benchmark, the authors manually annotated the vulnerable versions associated with 167 CVE entries across seven widely used Java projects. After automatically retrieving patch and version information from public repositories, each vulnerability instance underwent meticulous manual auditing, including static code analysis of pre and post patch versions to determine vulnerability status. This manual review was particularly critical for correcting inaccuracies and filling gaps where metadata from the NVD was incomplete or erroneous. Although the paper does not explicitly use the term "curation," the manual refinement of vulnerability metadata aligns with this thesis's operational definition

of manual curation. VERJava exemplifies a vulnerability-focused tool that incorporates rigorous manual refinement of metadata to support benchmarking and evaluation. The dataset represents a static snapshot, drawing primarily from GitHub repositories for code commits and the NVD for vulnerability metadata.

Manual Curation of License Identification Rules in Ninka

Higashi et al. (2023) present Ninka, a rule-based license identification tool that employs manually constructed regular expressions and curated phrase sets to detect OSS licenses in source code headers and license files. Ninka matches license declarations against a static set of hand-crafted license sentences and phrases, requiring human curation to maintain rule coverage as new licenses and orthographic variants appear.

Although the system automates the detection step, the creation, validation, and updating of license rules after initial retrieval are entirely manual processes, aligning with this thesis's definition of manual curation. Specifically, Ninka performs manual validation, correction, and enrichment of its license detection rules based on curated examples. Specifically, Ninka exemplifies an early, labor-intensive approach to metadata refinement, relying heavily on human expertise for ongoing rule maintenance.

Manual Curation of Fix-Commits for SCA Evaluation

Ponta et al. (2019) compile a manually curated benchmark that maps known vulnerabilities (CVE) to their corresponding vulnerable and fixed commits across 205 open-source Java projects. Following automatic retrieval of candidate commits, each CVE-to-commit mapping undergoes meticulous manual validation to confirm that the identified commit effectively addresses the reported vulnerability. The curation process includes manual validation, correction of inaccuracies, and enrichment with additional metadata. Although the paper centers on dataset construction rather than the development of an SCA tool, the rigorous manual curation approach provides a high-quality reference for evaluating vulnerability-focused SCA tools, such as Eclipse Steady.

Manual Curation of Benchmark Data in Achilles

Dann et al. (2022) present Achilles, a manually curated benchmarking suite consisting of 2,505 real-world OSS test cases. Each case simulates common modifications, such as re-bundling or metadata removal, and is manually validated to establish a reliable ground truth regarding the presence or absence of vulnerabilities. Achilles thus exemplifies a manual curation effort focused on the validation and correction of vulnerability metadata.

4.4.2 Automatic Curation Practices

Having examined tools and datasets that rely on human validation and correction, this section transitions to fully automated approaches for refining license and vulnerability metadata. Tools in this category perform curation tasks entirely without human intervention, typically leveraging machine learning or rule-based systems to automate interpretation.

Automated License Metadata Curation in LiDetector

Xu et al. (2023) introduce LiDetector, a machine learning–based tool designed to detect open-source license incompatibility by automatically interpreting both standard and custom license texts. Unlike rule-based scanners, LiDetector leverages a bi-directional Long Short-Term Memory (bi-LSTM) model combined with Conditional Random Fields (CRF) to identify license term expressions, and a Probabilistic Context-Free Grammar (PCFG) model to infer rights and obligations from natural language. These automated inferences allow the tool to classify each license term's status as CAN, CANNOT, or MUST, based on its semantic context.

Although the paper primarily describes these processes as license comprehension and incompatibility detection, they directly align with this thesis's definition of automated metadata curation following automatic retrieval. For example, LiDetector automatically corrects inaccurate sentiment assignments (e.g., distinguishing permissive from restrictive clauses) and refines unstructured license text into structured, SPDX-aligned terms.

Notably, LiDetector does not incorporate any manual post-processing or validation layer. All curation steps, term extraction, attitude inference, and incompatibility resolution, are carried out programmatically. Evaluation results underscore its effectiveness: the tool achieved 93.28% precision in license term identification and 91.09% accuracy in rights and obligations inference, using static datasets collected from GitHub and tldrlegal. LiDetector exemplifies a fully automated curation pipeline, transforming heterogeneous and often ambiguous license texts into structured, canonical representations suitable for compliance analysis.

Automated Vulnerability Metadata Refinement in V1SCAN

Woo et al. (2023) introduce V1SCAN, a tool designed to detect 1-day vulnerabilities in reused C/C++ open-source components. To mitigate the limitations of traditional version, and code-based approaches, V1SCAN integrates Locality Sensitive Hashing (LSH) with deep classification techniques to categorize reused code into "exactly reused," "changed," or "unused" categories. It then filters out vulnerabilities not relevant to the reused portions. Although the term "curation" is not explicitly used, the process of filtering unused vulnerable code, correcting

mismapped vulnerabilities, and normalizing reused code aligns closely with this thesis's operational definition of automated curation.

V1SCAN's curation activities are fully automated after initial retrieval. They involve automatic validation (e.g., checking whether vulnerable code is actually reused), correction (e.g., filtering out vulnerabilities patched by backporting), and enrichment (e.g., refining vulnerability evidence by combining version and code level information). Thus, V1SCAN exemplifies an automated, vulnerability-focused SCA tool that embeds curation implicitly within its high-precision detection process.

Automated Vulnerability Metadata Curation in Vul4Java

Wang et al. (2024) present Vul4Java, a two-stage vulnerability detection method designed to improve the accuracy of vulnerability metadata in Java OSS. In the first stage, the system constructs mapping relationships between third-party library binaries and publicly disclosed CVE entries by aggregating data from sources such as the NVD. The second stage involves validating these associations through static analysis and patch verification. Specifically, the tool generates structure-aware call graphs (SACGs) that integrate class hierarchy analysis and variable point analysis to more comprehensively trace invocation paths and identify potentially vulnerable functions.

Patch verification is conducted by extracting vulnerability-related functions and comparing their pre and post patch code features using structural and logical similarity metrics. This process helps detect whether a vulnerability is still present or has been patched in a given OSS binary.

Although the term "curation" is not explicitly used, Vul4Java's pipeline clearly aligns with this thesis's definition of automated curation, which involves post-retrieval validation, correction (e.g., filtering false positives), and enrichment (e.g., refining vulnerable function detection). All refinement steps are automated and operate on a static dataset comprising 373 OSS binaries and 1,203 vulnerability records sourced from Maven Central, NVD, and GitHub. The tool achieves a notable F1 score of 0.779, outperforming traditional tools such as Dependency-Check and OpenSCA. As such, Vul4Java exemplifies a vulnerability-focused SCA tool that embeds automated semantic refinement into its detection pipeline.

Automated License Metadata Curation in FOSSLT

J. Zhao et al. (2023) propose FOSSLT, a license identification model optimized for large-scale, mixed-source software systems. The tool enhances prior approaches like Ninka and Nomos by introducing a two-part architecture: a text extraction module that reduces redundancy by isolating license-relevant segments in license, copyright, and source files; and an identification module that applies

a novel N-Simhash matching algorithm. This dual strategy integrates Simhash with an N-gram language model to improve sensitivity to word order and suppress noise, enabling FOSSLT to match extracted license texts against canonical SPDX entries with higher precision and recall.

Although the authors do not explicitly use the term "curation", FOSSLT performs fully automated license metadata refinement. It validates extracted texts through fingerprint similarity matching, corrects mismatches by filtering out noise and irrelevant patterns, and enriches the metadata by mapping it to structured SPDX identifiers. These operations match the thesis's operational definition of automated curation. The tool runs without manual intervention after data ingestion.

FOSSLT was evaluated on a static dataset of over 3,000 license-containing files (and 300 non-license files) sourced from Linux distributions. The dataset spanned over 80 license types. Compared to tools like Ninka, Fossology, and Go License Detector, FOSSLT achieved the highest precision (99.05%), recall (87.94%), and a notable F1 score of 0.9316, while also exhibiting the shortest processing time. These results highlight FOSSLT as a license-focused tool that performs automated license metadata curation through rule-optimized and machine-assisted refinement.

Automated Metadata Refinement in Movery

Woo et al. (2022) present Movery, a vulnerability clone detection tool that precisely identifies vulnerable code clones (VCCs) in modified OSS components. Movery performs fully automated metadata refinement through the generation of semantic signatures based on historical CVE patches, which are retrieved from public sources like the NVD. These signatures capture core vulnerability and patch patterns through a two-stage process: (1) generating vulnerability and patch signatures using essential and dependent lines from both disclosed and older vulnerable functions; and (2) automatically validating reused code by checking for matches with these signatures, while correcting mismatches and distinguishing patched from vulnerable code. This detection process is fully automated after data retrieval and does not require manual verification.

Movery thus exemplifies automated curation as defined in this thesis: it validates, corrects, and refines vulnerability metadata automatically to account for OSS modifications and syntax variations. It operates on a static snapshot composed of 4,219 CVE patches and tested against 10 real-world C/C++ systems, achieving 96% precision and recall.

Automated Vulnerability Metadata Curation in CVEfixes

Bhandari et al. (2021) present CVEfixes, a fully automated tool and dataset for mining CVE records from the NVD, retrieving vulnerability-fixing commits from linked repositories (e.g., GitHub, GitLab, Bitbucket), and enriching the data with detailed metadata. The system extracts and structures data at multiple abstraction levels, commit, file, method, and automatically augments it with security and code-related metrics such as CWE types, CVSS scores, and programming languages. According to the authors, "the CVEfixes collection tool automatically fetches all available CVE records from the NVD, gathers the vulnerable code and corresponding fixes from associated open-source repositories, and organizes the collected information in a relational database". They explicitly describe the process as involving automatic collection and curation. The tool enables repeated extraction aligned with daily updates from the NVD but requires re-execution to refresh the static snapshot, maintaining a reproducible structure without human curation. CVEfixes thus aligns with the thesis's definition of automated curation, combining validation, correction, and enrichment processes without manual intervention.

Automated Vulnerability Metadata Curation in Holmes

Wu et al. (2024) introduce Holmes, a fully automated component designed to identify libraries and their ecosystems affected by known vulnerabilities. Holmes collects structured and unstructured evidence from multiple sources, including NVD metadata, GitHub repositories, and package registries such as Maven, PyPI, and Go. It applies a learning-to-rank approach that uses BM25 scoring and a LambdaMART model to prioritize candidate libraries associated with a given CVE. The entire process of evidence retrieval, feature extraction, and ranking is conducted without human intervention, aligning with this thesis's definition of automated metadata curation.

Although Holmes is fully automated, the authors acknowledge that industrial SCA vendors such as GitHub, Snyk Open Source, and Veracode continue to rely on manual curation by expert security teams to resolve inconsistencies across public vulnerability databases. Holmes is intended to complement this manual effort by providing scalable, evidence-based predictions that help identify gaps and inaccuracies. Therefore, while the broader ecosystem reflects hybrid practices, Holmes itself exemplifies automated curation of vulnerability metadata through machine learning–based evidence synthesis.

Automated Vulnerability Metadata Curation in VODA

VODA, introduced in the study by Kumar et al. (2024) represents a SCA tool designed to analyze and quantify the impact of vulnerable dependencies across

large open-source ecosystems. VODA performs fully automated metadata enrichment and validation by generating Software Bill of Materials (SBOM) using the CycloneDX format and matching components against known vulnerabilities from both the Sonatype OSS Index and the NVD.

The tool identifies affected components, determines the point in time at which a vulnerability was introduced, publicly disclosed, and fixed, and tracks whether a given project has resolved the issue. This process involves correcting inconsistencies in vulnerability metadata by reconciling SBOM entries with real-world fix commit information and standardizing the mapping between packages and CVE entries. Although the term "curation" is not explicitly used, the system clearly satisfies the thesis's operational definition of automated curation by performing metadata validation (e.g., CVE-to-component linking), correction (e.g., identifying unresolved vulnerabilities across versions), and enrichment (e.g., adding lifecycle metadata such as resolution times).

4.4.3 Hybrid Curation Practices

This section examines tools and components that combine both manual and automated methods to refine metadata during the SCA process. In hybrid approaches, automation may handle initial pattern matching, while human experts intervene to validate ambiguous results, correct misclassifications, or extend rule coverage. These systems aim to balance scalability with accuracy, and often embed mechanisms for user feedback, expert review, or community contributions.

Hybrid License Metadata Curation in DIKE

Cui et al. (2023) present DIKE, a tool designed to detect and resolve license conflicts in Free and Open Source Software (FOSS) through a combination of automated term extraction and manual expert validation. DIKE builds a *License-Terms-Responsibility* knowledge base by extracting 12 key license terms from 3,256 open source licenses using a Label-Specific Attention Network (LSAN) and classifying term responsibilities (e.g., permitted, required) using the ALBERT language model. The initial stage involves manual annotation of 319 license texts, reviewed by legal experts, which are then used to train and validate the machine learning models, establishing the hybrid nature of the curation pipeline. This curated knowledge base is subsequently matched against scan results from software projects to detect license conflicts and recommend solutions, such as code or license replacement. While the tool performs automated detection and matching at scale, the foundational data relies on human-curated labels and expert-reviewed annotations, aligning it with this thesis's definition of hybrid curation. DIKE's effectiveness is demonstrated through a large-scale evaluation of over 16,000 GitHub projects, highlighting the prevalence of license conflicts and the value of enriched license metadata.

Although the term curation is not explicitly used in the paper, the described methodology aligns closely with this thesis's operational definition. DIKE's approach represents a hybrid curation model, where initial extraction is performed algorithmically and subsequently verified by human annotators.

Hybrid Vulnerability Metadata Curation in PatchScout

Tan et al. (2021) present PatchScout, a ranking-based system designed to streamline the identification of security patches for disclosed OSS vulnerabilities. Instead of relying on strict matching approaches, PatchScout transforms the task into a learning-to-rank problem by evaluating semantic and structural correlations between vulnerability metadata (from CVE/NVD) and code commits across repositories (Tan et al., 2021). PatchScout automates several aspects of the metadata refinement process: it extracts vulnerability information (e.g., identifiers, locations, types, and descriptive texts), generates 22 correlation features, and trains a RankNet model to rank code commits in terms of patch relevance. These automated steps operationalize metadata validation and enrichment by identifying patch commits even when they lack explicit CVE references or are incorrectly tagged in the NVD.

Importantly, the final patch confirmation step is manual, requiring human reviewers to examine top-ranked commits to verify correctness. As the authors note, *"PatchScout also requires manual efforts to finally locate the security patches from the ranked code commits" and is designed to "help to locate more security patches and meet a balance between the patch coverage and the manual efforts involved"* (Tan et al., 2021). This hybrid approach of combining machine learning-based ranking with manual confirmation constitutes a hybrid curation methodology.

Hybrid Vulnerability Metadata Curation in Vulas

Ponta et al. (2018) present Vulas, a code-centric system developed at SAP for detecting, assessing, and mitigating vulnerabilities in open-source Java components. The authors explicitly state that the presented approach is implemented in the open-source tool Eclipse Steady, which serves as SAP's officially recommended SCA solution for Java applications. Unlike traditional tools that rely heavily on metadata, Vulas refines and corrects vulnerability-related metadata by statically and dynamically analyzing actual code usage. Specifically, it identifies vulnerable constructs by comparing Abstract Syntax Tree (AST) changes introduced in CVE fix commits, and then verifies whether those constructs are reachable in downstream applications using both static call graph analysis and runtime instrumentation.

Although the term "curation" is not explicitly used, the tool performs clear curation tasks as defined in this thesis. The automated components of Vulas validate

and correct vulnerability metadata by detecting reachable vulnerable code that is often missed or misclassified by metadata-only approaches. At the same time, the tool's internal knowledge base, which maps CVE entries to source-level code changes, is manually curated by experts, with the authors noting that maintaining such a resource is labor-intensive and not yet fully automatable. Thus, Vulas exemplifies a hybrid curation model that combines code-centric validation and enrichment with ongoing human input to maintain the accuracy and completeness of vulnerability metadata.

Preliminary Observations on Curation Trends

To synthesize the findings from the reviewed literature, Table 4.4 provides a structured summary of the curation types and focus areas across the analyzed SCA solutions. The table includes both standalone tools and modular components that perform or enable metadata curation. Resources such as CVEfixes, Achilles, and the vulnerability-fix dataset for Vulas are excluded, as they function primarily as benchmarks or empirical datasets rather than operational tools. The table highlights the diversity of curation approaches, manual, automated, and hybrid, and emphasizes the growing trend toward automation in vulnerability analysis workflows.

More broadly, the academic landscape reflects early and uneven patterns in how SCA tools implement metadata curation. Vulnerability-focused tools increasingly favor automated refinement techniques, while license-focused tools tend to rely on expert-crafted rules or hybrid pipelines. However, despite the apparent intent to refine metadata, the implementation details of these workflows, such as the logic for validation, correction, or enrichment, are often undocumented or only implied in broader descriptions of tool functionality.

In some reviewed studies, curation practices could not be meaningfully extracted due to a lack of transparency or limited functionality. For instance, while ASLA focuses on license detection via regular expressions and dependency graphs, it does not implement mechanisms for validating or refining metadata (Tuunanen et al., 2006). Similarly, OMP tool prioritizes structured documentation and traceability, but in the face of conflicting or missing license data, the approach defaults to exclusion rather than correction (Dyck et al., 2018). Also, the comparative evaluation by Imtiaz et al. (2021) focuses primarily on output-level metrics such as vulnerability counts and dependency coverage, rather than internal workflows that reflect curatorial decisions. Nonetheless, these studies were retained due to their value in highlighting key limitations in current SCA tools, the diversity of data sources used, and the variability in update frequency, insights that are crucial for identifying research gaps and shaping future development directions. Their inclusion also supports theoretical saturation, ensuring that this review adequately captures the spectrum of curation-related practices, both explicit and

implicit, found in academic literature.

Given the inconsistencies and underreporting observed in academic studies, further insights into operational curation practices now require turning to grey literature, including commercial SCA platforms and community-driven initiatives. These sources often provide more structured, scalable, and transparent workflows for metadata validation and enrichment, making them essential for a complete understanding of current best practices in SCA curation.

Table 4.4: Curation Practices across SCA Solutions

Tool / Component	Focus	Curation Type
VERJava	Vulnerability	Manual
Ninka	License	Manual
LiDetector	License	Automatic
V1SCAN	Vulnerability	Automatic
Vul4Java	Vulnerability	Automatic
FOSSLT	License	Automatic
Movery	Vulnerability	Automatic
Holmes	Vulnerability	Automatic
VODA	Vulnerability	Automatic
DIKE	License	Hybrid
PatchScout	Vulnerability	Hybrid
Vulas	Vulnerability	Hybrid

4.4.4 Curation Practices in Commercial SCA Tools

To address the gap in academic reporting, I conducted a grey literature review of four widely-used commercial SCA tools, supplemented by an analysis of two open and community-driven curation initiatives. The commercial tools analyzed include Snyk Open Source (n.d.), Black Duck (n.d.), Sonatype Nexus Lifecycle (n.d.), and JFrog Xray (n.d.). These tools were selected based on their frequent citation in academic literature, strong market presence, and the availability of detailed vendor documentation describing their scanning workflows, update mechanisms, and metadata enrichment processes.

In addition, this section includes non-commercial tools that support open and transparent curation practices. FOSSology (n.d.), developed by the Linux Foundation, focuses on license compliance and enables users to manually validate and correct license findings through an interactive interface. ClearlyDefined (n.d.), a community-led project, facilitates public contributions to improve the quality of open source metadata. These non-commercial platforms were included because they provide valuable insights into collaborative and reproducible approaches to

metadata curation, which complement the more closed practices of commercial SCA solutions.

Hybrid Metadata Curation in Snyk Open Source

Snyk Open Source (n.d.) employs a hybrid curation model that integrates automated data ingestion with manual validation and enrichment to ensure the accuracy of vulnerability metadata in its SCA platform. Its proprietary, triaged vulnerability database is continuously updated using inputs from standardized sources such as NVD (n.d.), and the GHSA (n.d.), as well as package registries, open forums, and internal research tools (Snyk Curation, n.d.). New vulnerability disclosures are monitored on a daily basis, leveraging both automated alerts and active human surveillance. Each advisory is subject to manual review by a dedicated security research team, which confirms the legitimacy of the issue, assigns accurate CVSS scores, defines affected and fixed version ranges, and enriches advisories with contextual summaries and remediation guidance.

Moreover, Snyk's designation as a CVE Numbering Authority (CNA) enables it to independently identify, disclose, and register vulnerabilities that may not yet exist in public databases, thereby contributing to the broader security ecosystem as explained by Danny Allan (2025). In contrast, according to the Snyk User Docs (n.d.), license metadata curation is more heavily automated but draws from a manually curated license catalog, enabling reliable license identification and policy enforcement. Collectively, these practices exemplify a hybrid curation methodology that corresponds with this thesis's definition of curation, namely, the validation, correction, and enrichment of vulnerability and license metadata.

Hybrid Metadata Curation in Black Duck

Black Duck (n.d.), Synopsys's SCA platform, employs a hybrid curation methodology that combines automated ingestion with expert-driven validation and enrichment. This model applies to both vulnerability data and license metadata, two core dimensions of SCA. Black Duck automatically retrieves data from public sources such as the NVD, GHSA, Google's OSV feed, and the SPDX License List. However, it does not rely solely on these feeds. Instead, a dedicated research team, the Cybersecurity Research Center (CyRC) curates this data by verifying its accuracy, correcting imprecise version ranges, identifying overlooked vulnerabilities, and resolving inconsistencies across sources.

For vulnerabilities, this curation culminates in the creation of Black Duck Security Advisories (BDSAs), which provide more refined and often earlier intelligence than public CVE entries, as explained in a blog by Black Duck (2022). These advisories include detailed metadata such as affected code paths, vulnerable version ranges, fixing commits, severity scores, and remediation guidance. The curation

is semi-automated: while data ingestion and tooling (e.g., commit differencing) are automated, the vulnerability analysis and advisory generation are expert-led. This approach reduces false positives and negatives, making the results more reliable for development teams.

In parallel, Black Duck curates a comprehensive license knowledge base, covering over 2,750 unique open source licenses as described in Black Duck KnowledgeBase (n.d.). License detection during scans is automated using text fingerprinting and multifactor heuristics, but the creation and maintenance of license definitions, including metadata on legal obligations and SPDX identifiers, are manual processes handled by Synopsys's legal and content experts. When a new or unknown license is encountered, it is reviewed and categorized before being added to the central database. This ensures both breadth (through automation) and accuracy (through expert review), aligning with best practices in metadata curation.

Thus, Black Duck demonstrates a robust hybrid model for curating automatically retrieved data: automated feeds provide coverage and speed, while expert oversight ensures semantic accuracy, legal reliability, and actionable insight. This dual process enhances the trustworthiness of vulnerability alerts and license compliance findings, embodying current best practices in the field of SCA.

Hybrid Metadata Curation in Sonatype Nexus Lifecycle

Sonatype Nexus Lifecycle (n.d.), part of the broader Nexus Platform, implements one of the most mature and structured hybrid curation models among commercial SCA tools. Vulnerability metadata is curated via *Nexus Intelligence*, Sonatype's proprietary data service, which aggregates signals from diverse sources, including the NVD (n.d.), GHSA (n.d.), OSV (n.d.), bug bounty disclosures, mailing lists, social media, and customer submissions. Rather than relying on these sources directly, Sonatype emphasizes correction and validation through its dedicated Data Research Team. As explained in the 'Sonatype Vulnerability Data' section of the official documentation, over 92% of public vulnerability data required correction upon expert review (Sonatype Vulnerability, n.d.). The validation process occurs in two phases: a *Fast-Track* curation layer provides an initial, rapid human review to verify affected components and assign metadata (e.g., vulnerable version ranges and CVSS scores), while a subsequent *Deep Dive* analysis refines the recorded information through source code inspection and remediation tracing. Vulnerabilities incorrectly attributed due to version mismatches, inaccurate impact scopes, or mislinked components are corrected or removed from the database.

Similarly, license metadata is curated with comparable rigor. Sonatype Sonatype Nexus Lifecycle scans both declared and observed licenses, validating package-level metadata against content-derived license evidence. The "Component Li-

cense Information" section of Sonatype License (n.d.) emphasizes that a key feature is the determination of the effective license, representing the legally binding terms based on all observed data. Legal experts categorize licenses into curated threat groups (e.g., *Copyleft*, *Permissive*, *Banned*), which drive compliance evaluations and policy enforcement.

Together, Sonatype Sonatype Nexus Lifecycle's approach exemplifies a comprehensive realization of metadata curation: automated ingestion is paired with expert-driven validation and semantic enrichment, resulting in a highly trusted metadata service that aligns closely with the thesis definition of curation as validation, correction, and enrichment.

Hybrid Metadata Curation in JFrog Xray

JFrog Xray (n.d.), part of the broader JFrog Platform, implements a hybrid metadata curation model for vulnerability information, emphasizing contextual validation and operational enrichment. Vulnerability data is automatically ingested from the NVD, which serves as a common baseline in the industry. However, JFrog's Security Research team independently curates this data by validating real-world exploitability, assigning proprietary severity scores (JFrog Severity Score), identifying affected functions or methods, and offering mitigation strategies, transforming raw CVE data into actionable insights, as explained in a blog by Ouzan and Shalom (2025).

These enriched insights are integrated directly into Xray's scanning engine via its Contextual Analysis feature, which evaluates whether a vulnerability is actually exploitable in a given runtime context (e.g., invocation of a vulnerable method or specific configuration states). This dynamic assessment significantly reduces false positives and improves triage relevance. Manual research conducted by JFrog's analysts further enables correction by downgrading, flagging, or suppressing vulnerabilities that are deemed non-impactful under specific deployment conditions. Regarding license metadata, Xray provides automated detection of licenses through manifest analysis and content-based license identification. It identifies SPDX-compliant licenses and supports the creation of custom license definitions, allowing users to override misclassified or unknown entries. These capabilities facilitate license compliance management by enabling organizations to define and enforce policies e.g., blocking copyleft licenses or requiring permissive ones (JFrog Xray License, n.d.).

However, it's important to note that while Xray offers robust tools for license compliance enforcement, these functionalities align more with policy enforcement mechanisms rather than metadata curation as defined in this thesis. According to the thesis's definition, curation involves the validation, correction, and enrichment of metadata. Xray's license management features primarily focus on applying

predefined policies to detected license information, without necessarily engaging in the manual validation or enrichment of the license metadata itself.

Nevertheless, JFrog (n.d.) frames policy enforcement as part of its broader curation offering, as reflected on its official website.

Xray's methodology exemplifies hybrid curation in practice for vulnerability data: automated aggregation is complemented by expert validation and contextual enrichment. For license data, while automated detection and policy enforcement are present, the aspects of manual validation and enrichment are less emphasized. The result is a reliable and practical metadata service that only partly matches this thesis's definition of curation.

Community-Driven Metadata Curation Platforms

In addition to proprietary, vendor-curated tools, platforms like FOSSology (n.d.) and ClearlyDefined (n.d.) represent community-driven approaches to metadata curation within SCA. FOSSology is a standalone license-focused SCA tool that automates license detection and enables user-driven validation, correction, and reporting. In contrast, ClearlyDefined is not a standalone analysis tool but an open curation platform that aggregates license metadata from upstream ecosystems and supports collaborative refinement through community contributions. While their technical scopes differ, both platforms prioritize transparency, standards alignment (e.g., with SPDX), and iterative metadata enrichment. Each platform's curation methodology is discussed in detail in the following subsections.

Manual License Metadata Curation in FOSSology

FOSSology (n.d.), an open-source license compliance tool maintained by the Linux Foundation, adopts a community-driven, human-in-the-loop approach to metadata curation, focusing exclusively on open-source licensing. Its architecture integrates automated license detection with manual validation and enrichment, aligning closely with the thesis definition of curation.

License detection in FOSSology is powered by specialized scanning agents, notably *Nomos*, a rule-based scanner that uses regular expressions and contextual heuristics, and *Monk*, which applies similarity metrics (e.g., Jaccard similarity) to compare file content against a curated library of known license texts. These scanners ingest license indicators from package files and source code, producing candidate matches with associated confidence levels. Detected licenses may be specific (e.g., MIT, GPL-2.0) or generic (e.g., "UnclassifiedLicense") if the system cannot definitively classify a finding, prompting human review.

The core curation process occurs after scanning, when users validate and correct

the automated findings via FOSSology's interactive web interface. Reviewers can override misclassified results, mark false positives, and assign more precise license identifiers based on contextual reading. Importantly, users can also introduce new or custom license definitions directly into the system, enriching the internal license database for future reuse as explained in FOSSology Workflow (n.d.). This collaborative enrichment mechanism not only corrects current scans but improves recognition capabilities over time.

Unlike commercial SCA tools with centralized research teams (e.g., Sonatype), FOSSology distributes curation responsibility to end users. Its interface supports bulk actions and highlighted text views to facilitate large-scale manual review, which is essential for organizations conducting internal license audits or integrating FOSSology into broader governance workflows. While the tool lacks built-in vulnerability scanning, it excels in transparency and customizability. FOSSology can export curated results as SPDX-compliant SBOM, enabling interoperability with external compliance and security tooling.

In summary, FOSSology exemplifies manual license metadata curation: it automates the discovery of license text but requires human expertise for validation and correction, and supports enrichment through community-driven contributions. This design makes it particularly suitable for teams seeking fine-grained control over license metadata and emphasizes curation as an active, participatory process.

Community-Driven Metadata Curation in ClearlyDefined

ClearlyDefined (n.d.) is not a conventional SCA tool but rather an open data curation platform designed to improve the completeness, consistency, and clarity of metadata for open source components, particularly with respect to licensing. Unlike commercial SCA tools such as Snyk Open Source, Sonatype Nexus Lifecycle, or Black Duck, which rely on proprietary knowledge bases curated by internal research teams, ClearlyDefined adopts a decentralized, community-driven approach to metadata curation.

According to Jeff McAffer (2019), ClearlyDefined continuously harvests component metadata from upstream ecosystems and applies automated scanning using tools like *ScanCode*, *FOSSology*, and *Licensee*. These tools extract raw metadata, including declared and discovered licenses, copyright notices, and source code locations, into structured "definitions" for each package. Crucially, ClearlyDefined explicitly distinguishes between uncurated scan results and curated metadata. Community contributors validate and improve these results by submitting GitHub pull requests to the project's public repository. As described in the ClearlyDefined Curation (n.d.) of the ClearlyDefined website, each contribution undergoes community peer review, is aligned with SPDX-like metadata standards and

project guidelines, and is version-controlled via GitHub to ensure transparency and traceability.

This model reflects the thesis's definition of curation as a process of validation, correction, and enrichment. Validation occurs when contributors confirm scanner findings against license texts. Corrections are made when tools misidentify or miss license information (e.g., replacing "NOASSERTION" with a valid SPDX ID). Enrichment includes the addition of missing metadata. A notable feature, highlighted by ClearlyDefined (n.d.) is ClearlyDefined's emphasis on upstream contribution, encouraging curators to open pull requests in the original codebase when metadata is absent or needs clarification.

ClearlyDefined's open model stands in contrast to proprietary workflows such as Sonatype Nexus Lifecycle's Fast-Track and Deep Dive or Synopsys's internal advisory generation. Instead, curation responsibilities are distributed across a global network of developers, legal professionals, and compliance experts. This ensures that the resulting metadata is not only high-quality but also publicly accessible via API and exportable in SPDX-aligned formats. According to Nick Vidal (2024), GitHub integrates ClearlyDefined's curated dataset to enhance its license compliance solutions

ClearlyDefined shows how community-driven efforts to improve metadata can work alongside official SCA tools, helping connect open collaboration with more formal compliance processes.

4.5 Data Sources Used in Analyzed SCA Solutions

An essential aspect of evaluating curation practices is understanding the data sources that SCA solutions, both tools and components, rely on. In this context, a data source refers to the repository or knowledge base against which the tool compares software dependencies or license declarations. These sources may include standardized vulnerability databases (e.g., NVD), curated license reference datasets (e.g., SPDX license list), or vendor-maintained metadata feeds. To keep the analysis manageable and focused, only the primary data sources explicitly mentioned were considered.

This section draws on information extracted from both academic and grey literature. A detailed breakdown of the primary data sources used by the analyzed SCA solutions is presented in Table 1, located in Appendix A. Initially, the review identified 29 unique SCA solutions from academic sources, including both standalone tools and supporting components. Upon incorporating tools from grey literature, specifically Sonatype Nexus Lifecycle and JFrog Xray, the total num-

ber of analyzed SCA solutions increases to 31. It is important to note that each tool is counted only once in the final total, even if it appears in both academic and grey literature sources. This ensures consistency and avoids duplication in the data source analysis.

4.5.1 Distribution of Data Sources

The primary data sources used across the analyzed SCA tools and components fall into six major categories:

NVD remains the most widely used public source for vulnerability detection, serving as the foundational registry for 8 tools (NVD, n.d.). These tools use NVD to compare known CVE entries against software dependencies. While often combined with other feeds in practice, NVD remains the core source in many academic SCA tools.

Proprietary and vendor-specific data sources were cited as the primary comparison dataset in 9 tools. These tools primarily are commercial solutions such as Snyk Open Source (n.d.), Black Duck (n.d.), Sonatype Nexus Lifecycle (n.d.), and JFrog Xray (n.d.), supplement public databases with internally curated advisories, bug bounty findings, forums, and product-specific feeds. These sources underpin hybrid enrichment and private intelligence workflows. Importantly, tools appearing in both academic and commercial contexts (e.g., Snyk Open Source, Black Duck) are counted only once in this analysis under their commercial configurations.

The SPDX License List is the primary license metadata source in 5 tools, especially those focused on license classification and compliance (SPDX, n.d.). The SPDX list provides standardized identifiers and templates, enabling structured license detection from source code, manifests, or binary signatures.

GHSA were referenced by only 1 tool as it's primary data source (GHSA, n.d.). GitHub's advisory system enables linking vulnerabilities to commits, patches, and affected packages.

The OSV database, maintained collaboratively by Google and the open-source community, was used as the primary source by only 1 tool (OSV, n.d.). OSV focuses on package-aware vulnerability entries and is particularly relevant for SBOM-based scanners.

Other sources, used by 7 tools, include a range of manually curated or domain-specific resources. These encompass handcrafted rule sets (e.g., regular expression–based classifiers), compatibility matrices, library-license mappings, and specialized package registries. While not standardized, these resources often offer valuable heuristics or metadata extraction capabilities in license or vulnerability-focused tools.

This distribution is illustrated in Figure 4.5, which visualizes how different focus areas, vulnerability-focused, and license-focused correlate with the use of specific data source categories.

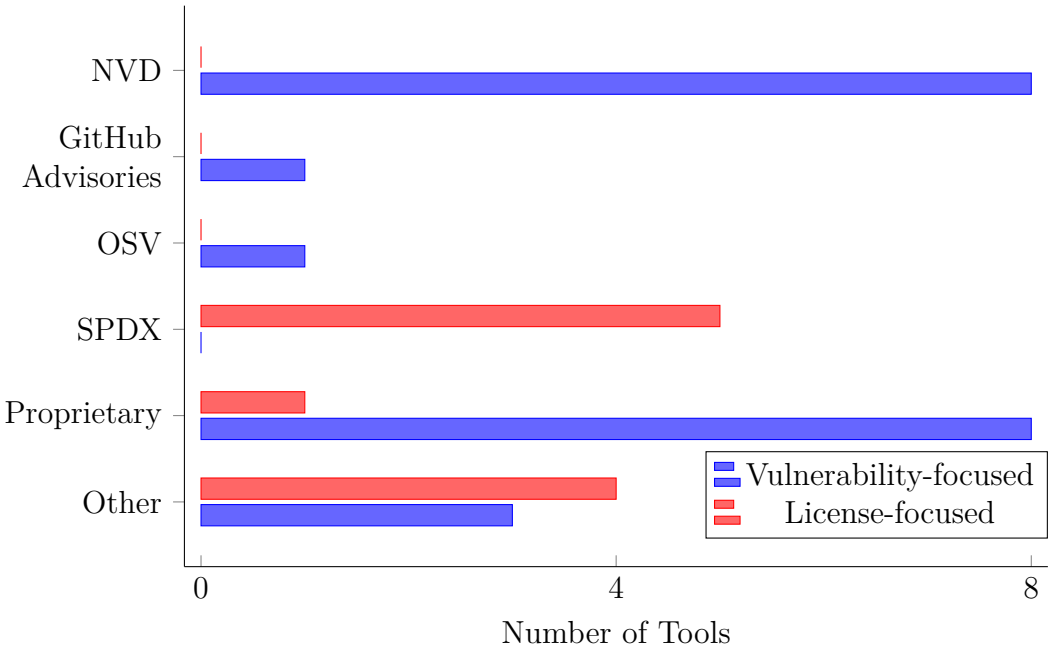


Figure 4.5: Distribution of Data Source Usage by Tool Focus Area

4.5.2 Data Source Trends by Focus Area

Clear patterns emerge when analyzing the primary data sources used by SCA tools, categorized by their core focus: vulnerability detection or license compliance.

Vulnerability-focused tools predominantly rely on the NVD, which serves as the foundational source for known CVE entries in **8 tools**. Some also incorporate alternative vulnerability sources: GHSA, OSV (used by OSV Scanner), or curated mirrors like Anchore's database. Additionally, **8 vulnerability tools** rely on proprietary vulnerability datasets, including Snyk Open Source, Black Duck, and VULAS (Ponta et al., 2018). These tools often integrate enriched or product-specific intelligence with public feeds.

License-focused tools center their analysis on structured corpora like the SPDX License List (used by 5 tools, including ScanCode, n.d. and FOSSology, n.d.), as well as manually curated resources. For instance, DIKE relies on an internal knowledge base, and Ninka use handcrafted rule sets (Cui et al., 2023). In total, **4 license-focused tools** fall under the "Other" data source category, reflecting domain-specific detection logic rather than external feeds.

Overall, the analysis shows that standardized datasets like the NVD and SPDX License List remain essential to academic and open-source tools, providing transparency and interoperability. In contrast, commercial offerings increasingly depend on proprietary data, which supports richer insights but raises concerns about transparency, access, and reproducibility. These implications are further explored in the Discussion section.

The next section explores the update frequencies of these data sources, contrasting continuous feeds with static or manually curated snapshots, and discusses how this impacts the timeliness and accuracy of SCA tool outputs.

4.6 Update Frequency of Data Sources in SCA Solutions

Update frequency in SCA refers to how often a tool synchronizes or refreshes its underlying data sources, such as vulnerability databases, license repositories, or curated metadata, used to analyze software dependencies for known issues.

Update frequency directly impacts how quickly SCA tools can respond to newly disclosed vulnerabilities or license changes. Tools with continuous update mechanisms can promptly reflect the latest information, enabling faster detection and reducing the exposure window. In contrast, tools that rely on static snapshots or manual updates may experience delays in identifying emerging issues, potentially leading to outdated or incomplete results.

In this study, SCA tools are categorized based on their update mechanisms into three primary types, as summarized in Table 4.5. These classifications reflect how each tool acquires and maintains its metadata for vulnerability and license analysis.

To ensure consistency and avoid duplication, if a tool is described in both academic literature and commercial documentation, only the update frequency reported in the commercial source is considered. This approach prioritizes the most detailed and context-specific description of the tool's behavior.

Figure 4.6 presents the distribution of update frequencies grouped by focus area (vulnerability, or license). It reveals that vulnerability-focused tools are more likely to adopt continuous update pipelines, whereas license-focused tools are more reliant on manual or static data curation.

As summarized in Table 2 in Appendix B, academic tools such as LiDetector (Xu et al., 2023), and VERJava (Sun et al., 2022) rely on static datasets, while others like ScanCode (n.d.), findOSSLicense (Kapitsaki et al., 2022), and ASLA (Tuunanen et al., 2006) are manually updated by developers. In the commercial

Table 4.5: Update Frequency Types in SCA Solutions

Type	Definition	Example Tool
Manual	Data is maintained by human experts without automation. New information, such as fix commits or license rules, must be manually added or verified.	DIKE
Static	Dataset is obtained as a one-time snapshot. There is no built-in mechanism for automated updates unless manually re-run.	VERJava
Continuous	Systems automatically synchronize with external data sources (e.g., NVD, GitHub Advisories) on a frequent or real-time basis without user intervention.	Black Duck

domain, platforms such as Snyk Open Source (n.d.), JFrog Xray (n.d.), and Sonatype Nexus Lifecycle (n.d.) integrate multiple data feeds and continuously update their vulnerability and license intelligence. These findings underscore the importance of update mechanisms as a core capability in determining tool responsiveness and real-world applicability.

Future research should provide more transparency around update frequency and clearly distinguish whether synchronization is continuous, manual, or hybrid. This information is vital for assessing tool reliability, evaluating real-time threat coverage, and ensuring reproducibility in empirical software engineering.

4.7 Metadata Curation Practices Beyond the SCA Domain

While SCA tools focus on curating metadata related to software components, such as licenses, and vulnerabilities, parallel innovations in other domains offer valuable lessons in metadata curation methodologies. Two notable examples include NVIDIA's NeMo Curator and the QURATOR project. NeMo Curator, developed by NVIDIA, is an open-source, GPU-accelerated pipeline for preparing high-quality, large-scale datasets for AI model training (NVIDIA, n.d.). QURATOR, on the other hand, is a research-driven curation platform focused on enriching and structuring unstructured textual content for sectors such as journalism, healthcare, and cultural heritage (Rehm et al., 2020).

Both systems feature modular, extensible architectures that enable reproducible, auditable, and domain-specific data refinement. This section presents a con-

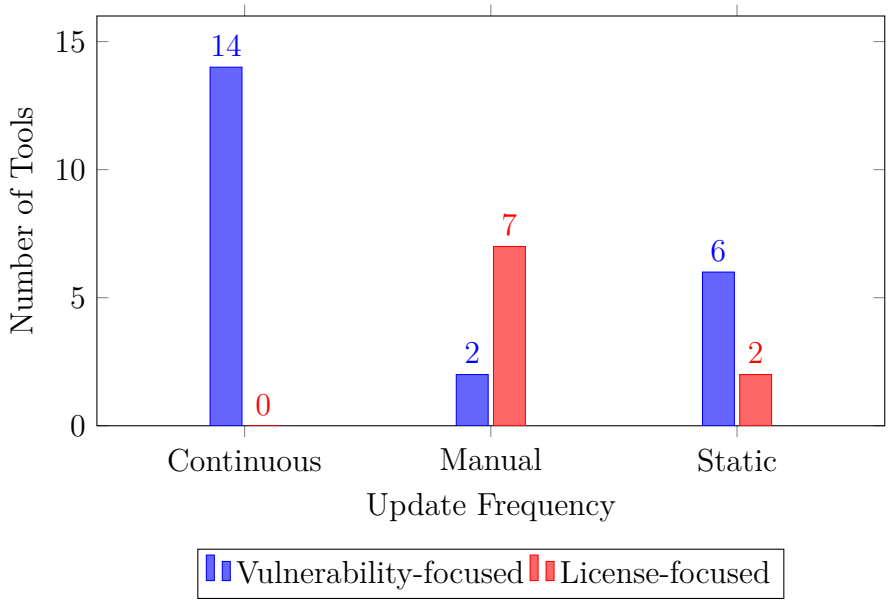


Figure 4.6: Update Frequency of Data Sources Used by SCA Tools

densed overview of the methodologies employed in NeMo Curator and QUR-ATOR, examining their relevance to advancing transparency and traceability in the curation workflows of contemporary SCA tools such as Snyk Open Source (n.d.), Black Duck (n.d.), and others.

NeMo Curator: End-to-End Metadata Curation Methodology

In the domain of large-scale Artificial Intelligence (AI) dataset preparation, NVIDIA's NeMo Curator presents a well-documented, open-source example of reproducible metadata curation. NeMo Curator offers a GPU-accelerated, modular pipeline designed to process web-scale datasets for training Large Language Model (LLM) and multimodal systems as described in a blog by Jennings et al. (2023). It enables users to clean, validate, deduplicate, and annotate large corpora in a transparent and auditable way. Each stage of the pipeline is implemented as a plug-and-play module operating on a standardized data format JSON Lines (JSONL), allowing consistent logging and traceability across billions of tokens.

The NeMo Curator pipeline begins with large-scale data ingestion from public sources such as Common Crawl, Wikipedia, and arXiv. The extracted raw data is then standardized into a unified JSONL format, allowing for modular processing and maintaining traceability throughout the pipeline. Following standardization, privacy and toxicity filtering modules are applied to detect Personally Identifiable Information (PII) and flag potentially harmful or offensive content. An essential next step is deduplication, which removes redundant content using ex-

act, fuzzy, or semantic similarity techniques to ensure that the dataset remains diverse and non-repetitive. After deduplication, a suite of heuristic and machine learning-based validation filters is used to evaluate aspects such as content quality, coherence, and language classification, thereby enabling the exclusion of low-quality or irrelevant documents. Finally, each curated document is enriched with metadata, including language tags, domain labels, and quality scores, resulting in a structured and self-descriptive dataset suitable for training large language models.

The diagram in Figure 4.7 presents an overview of this pipeline and was constructed based on descriptions provided in Jennings et al. (2023), the official NVIDIA GitHub documentation (n.d.), and the NeMo Curator blog by Nguyen (2024).

NeMo Curator's reproducibility is supported by deterministic modules and YAML-based configuration files, as described in official NVIDIA GitHub documentation (n.d.), enabling users to re-run the pipeline on different data sources using the same configuration or to audit how specific datasets were curated. Additionally, NeMo Curator supports integration with external tools such as spaCy for natural language processing, Apache Spark for large-scale data handling, and pretrained transformer models from the NeMo framework for tasks like toxicity detection and domain classification. These features collectively establish a transparent, extensible, and auditable curation process that contrasts with the typically proprietary workflows used in SCA tools.

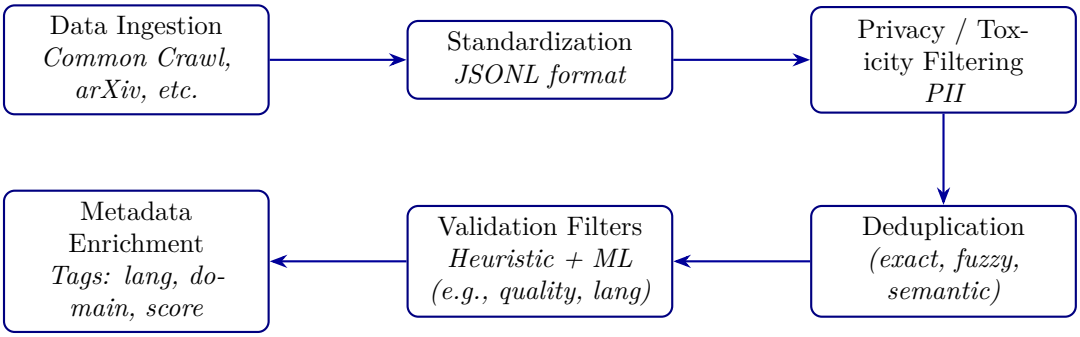


Figure 4.7: Overview of the NeMo Curator pipeline

Insights from the QURATOR Project

QURATOR is a collaborative research platform aimed at supporting content and data curation across sectors such as cultural heritage, journalism, healthcare, and enterprise communication (Rehm et al., 2020). Unlike SCA tools that curate metadata about software components, QURATOR focuses on curating unstructured textual content, such as news articles, medical records, and internal enterprise documents, by enriching it with semantic metadata and refining it for

downstream use. Its curation process is implemented as a sequence of AI-powered microservices that follow a consistent and auditable workflow.

The curation methodology begins with preprocessing modules that validate and standardize raw inputs: duplicate texts are removed, document structures (e.g., headlines, sections) are recognized, and language identification is performed. These steps ensure consistency across heterogeneous data sources. Enrichment modules then curate the content by extracting named entities (people, places, organizations), tagging temporal expressions, and detecting events and relations within the text. The curated data is further refined using knowledge graphs and reasoning engines to increase its semantic depth and relevance. Finally, the platform generates value-added outputs through summarization, translation, and storytelling services tailored to specific domains (Rehm et al., 2020).

The output of this curation process is content that is machine-interpretable, semantically annotated, and adaptable for domain-specific applications. Without curation, such content would remain unstructured and difficult to reuse, hindering automated processing, personalization, and knowledge extraction. Through this layered architecture, QURATOR delivers a modular and transparent curation pipeline for textual data. Each service contributes to validating, enriching, or transforming the data in a reproducible way. Although QURATOR is not designed for software metadata, its transparent and modular curation framework can inform improvements in how SCA tools validate, annotate, and refine component metadata.

5 Discussion

This chapter synthesizes and interprets the findings presented in Chapter 4, with the aim of answering the primary research questions defined in Chapter 3. These questions sought to uncover current practices for metadata curation in SCA tools, analyze how such practices are documented in academic and grey literature, evaluate the methodologies used in assessing these tools, and derive comparative insights from adjacent domains. The discussion is structured thematically around these objectives.

5.1 Curation Practices in SCA Tools (RQ1)

The first research question asks: *What are current practices for curation of automatically retrieved data during SCA, like copyright or licensing information?* To answer this, the study examined both academic sources and industry documentation, seeking to identify how SCA tools refine automatically collected data. This overarching question is further decomposed into two sub-questions: (1) What practices does the scientific and gray literature describe? (addressed in Section 5.2); and (2) How have existing tools been evaluated? (addressed in Section 5.3).

Drawing on insights from peer-reviewed studies and an analysis of six commercial and community-driven platforms, this section identifies three overarching categories of metadata curation strategies emerged: manual, automated, and hybrid. These categories reflect distinct approaches to validating, correcting, and enriching metadata retrieved during SCA workflows. These categories reflect the strategies used to validate, correct, and enrich metadata related to vulnerabilities or licenses in SCA tools.

Manual curation remains a foundational practice in SCA tool development, particularly for constructing high-quality benchmarks. These methods rely on human expertise to validate, correct, and enrich metadata through static code review, handcrafted rule sets, or direct CVE-to-commit mapping. While less scalable than automated approaches, manual curation offers precise control and

transparency, with decision logic often explicitly documented.

Such methods are typically applied to static datasets and lack continuous update mechanisms, limiting adaptability in fast-evolving ecosystems. In the reviewed literature, manual curation is most common in early-stage or evaluation-focused tools. Notable examples include VERJava (Sun et al., 2022), Ninka (Higashi et al., 2023), and the curated dataset by Ponta et al. (2019), all of which demonstrate the continued value of manual refinement for ensuring metadata accuracy in vulnerability assessment and license classification.

Automated curation, increasingly used in modern SCA tools, refines metadata through algorithmic methods such as machine learning, semantic code matching, and probabilistic grammars. These approaches enable scalable and repeatable processing, often applied to large static datasets without human intervention.

While automated curation offers efficiency and speed, it often lacks transparency, making it difficult to interpret decisions in ambiguous cases, especially concerning in compliance-critical contexts. Compared to manual methods, automated pipelines function as "black boxes," limiting trust and reproducibility.

In the reviewed literature, automated curation is more common in vulnerability-focused tools, reflecting the need to scale with fast-evolving ecosystems. By contrast, license-focused tools still favor manual or hybrid models, given the legal complexity and contextual sensitivity of license interpretation.

Hybrid curation combines automated refinement with human oversight to balance scalability and accuracy. In these systems, models handle routine metadata processing, while experts review edge cases, verify ambiguous matches, or maintain curated datasets, particularly where legal or technical precision is critical.

This approach offers more contextual accuracy than full automation but requires more effort and slower update cycles, especially when manual validation is frequent. Hybrid strategies are often used in tools dealing with complex metadata, such as patch validation or license obligations, where automation alone may fall short.

Though less common than automated methods, hybrid curation appears in both vulnerability and license-focused tools when misclassification risks are high or when training data needs expert input. This reflects a growing recognition that human–machine collaboration can enhance both reliability and interpretability in SCA metadata workflows.

5.2 Curation in Scientific and Grey Literature (RQ2)

This section addresses the second research question: *What practices does the scientific and grey literature describe?* It focuses on how metadata curation is framed, described, or embedded in scholarly and non-academic sources.

In scientific literature, metadata curation is rarely labeled as such. Instead, terms like "validation," "benchmarking," or "refinement" are used, often without clarifying whether human oversight or automation is involved. Papers on tools such as PatchScout (Tan et al., 2021), Vulas (Ponta et al., 2018), and DIKE (Cui et al., 2023) implicitly describe curation-aligned practices, yet lack consistent terminology or detailed workflow explanations. Manual curation steps are more traceable in these studies, especially when used to build benchmarks or correct license misclassifications. However, automated methods, though increasingly common—are often described only at a high level, with little insight into decision logic or enrichment strategies.

In contrast, grey literature provides more structured and transparent accounts of curation. Commercial tools like Snyk Open Source (n.d.), Black Duck (n.d.), and Sonatype Nexus Lifecycle (n.d.), describe multi-step processes involving real-time data ingestion, expert review, and semantic enrichment. These vendors often present curation as a product feature, with dedicated teams, feedback mechanisms, and version control. Similarly, community-driven initiatives such as ClearlyDefined (n.d.) and FOSSology (n.d.) emphasize open contribution models and auditable metadata improvements, reinforcing trust and reproducibility.

Overall, the academic and grey literature reflect different emphases. Academic work highlights technical innovation but offers limited insight into continuous metadata refinement. Grey sources prioritize operational clarity, emphasizing the traceability and responsiveness of curation workflows. This contrast underscores the opportunity for academic studies to adopt clearer terminology and document curatorial practices more explicitly to increase their practical relevance and comparability.

5.3 Evaluation Practices for SCA Tools (RQ3)

This section addresses how SCA tools are evaluated in both academic and grey literature, particularly with regard to metadata curation, defined here as the validation, correction, and enrichment of retrieved metadata.

Academic studies primarily evaluate SCA tools using detection metrics such as precision, recall, and F1 score, often against manually curated datasets. Tools

like VERJava (Sun et al., 2022), PatchScout (Tan et al., 2021), and Vulas (Ponta et al., 2018) use ground truth mappings (e.g., CVE-to-commit) to benchmark scanner accuracy. However, these evaluations rarely assess how metadata is refined post-retrieval. As a result, the effectiveness of internal curation steps, such as false positive filtering or enrichment, is underreported, limiting insight into reproducibility and robustness.

Comparative academic studies, such as Imtiaz et al. (2021), focus on output coverage but typically do not analyze how differences in metadata quality influence results. This lack of granularity makes it difficult to assess how curated data contributes to overall tool performance.

In contrast, grey literature, especially from commercial tools, offers more detailed descriptions of curation workflows. These include update frequencies, validation stages, and expert involvement, often communicated through technical blogs and documentation. While these sources emphasize practical improvements such as reduced false positives, they rarely use open or reproducible metrics. For example, Sonatype reports correcting over 92% of public vulnerability records, but does not disclose evaluation methods (Sonatype Vulnerability, n.d.).

Community-driven platforms like ClearlyDefined take a different approach by enabling peer-reviewed curation through transparent logs, version control, and community agreement (ClearlyDefined Curation, n.d.). This promotes reproducibility, though it may lack formal statistical benchmarks.

Overall, academic evaluations offer standardized but limited views of tool performance, while grey literature highlights operational transparency without consistent metrics. Bridging these approaches, by explicitly measuring how curation improves detection accuracy or compliance outcomes, would improve the relevance and comparability of future SCA tool assessments.

5.4 Metadata Curation Beyond SCA (RQ4)

The fourth research question asks: *What lessons can be drawn from metadata curation practices in other fields?* To address this, two non-SCA systems, NeMo Curator by NVIDIA (n.d.) and the QURATOR project by Rehm et al. (2020), were selected for in-depth analysis. Both represent state-of-the-art, domain-specific platforms for curating large-scale data, and they offer a stark methodological contrast to existing SCA tools.

What distinguishes these systems is not merely their technical sophistication but their principled commitment to modularity, transparency, and reproducibility, qualities that are frequently lacking in both academic and commercial SCA workflows. Their curation pipelines are decomposed into discrete, inspectable

stages; their actions are consistently logged; and their design explicitly supports reusability, versioning, and configuration management. These properties make them not only auditable but also testable using well-defined evaluation metrics.

NeMo Curator

NeMo Curator was developed by NVIDIA to prepare large-scale, high-quality datasets for training LLM (NVIDIA, n.d.). It performs end-to-end metadata refinement across billions of web-scraped documents using a fully auditable pipeline. Curation tasks are modularized into standardized stages such as extraction, cleaning, quality validation, deduplication, and enrichment (Jennings et al., 2023). Each module logs its decisions and transformations in metadata fields, enabling detailed traceability.

What sets NeMo apart from most SCA tools is its emphasis on reproducibility. All steps are parameterized using YAML configuration files, and the entire process is version-controlled. This ensures that the same dataset can be curated identically across environments or re-executed with changes isolated to specific modules. Unlike SCA platforms, where curation decisions (e.g., version mapping or CVE tagging) often remain opaque, NeMo's design makes the logic of each refinement step fully accessible to developers and researchers.

Moreover, NeMo introduces explicit evaluation mechanisms. Data quality is scored using heuristic and machine learning–based filters (e.g., language consistency, coherence, toxicity). These quality annotations are embedded as metadata, allowing downstream applications to filter or prioritize data according to use case–specific thresholds. Such quality metrics are largely absent in SCA curation, where no standard exists for measuring the trustworthiness of license or vulnerability metadata.

QURATOR: Modular AI-Driven Curation

The QURATOR project presents a similarly modular and semantically rich approach to metadata curation, tailored to domains like journalism, cultural heritage, and healthcare (Rehm et al., 2020). It combines rule-based validation with AI-powered enrichment, extracting named entities, semantic relationships, and temporal information from unstructured text.

Unlike typical SCA tools, QURATOR applies a layered architecture where each microservice contributes to a specific aspect of curation, ranging from language detection and document structuring to ontology-based annotation and summarization. These microservices are chained into reproducible pipelines whose outputs are enriched and validated against domain-specific knowledge graphs.

Transparency is central to QURATOR's design: each service logs its transformations, and intermediate outputs can be audited or reused. Evaluation is similarly formalized. For example, entity recognition modules are benchmarked against gold standards, and semantic enrichment accuracy is measured using precision and recall, metrics that are rarely reported in SCA metadata refinement workflows.

Comparison with SCA Curation Models

Compared to NeMo and QURATOR, academic and grey literature SCA tools generally exhibit lower transparency, auditability, and methodological clarity. While tools like Sonatype Nexus Lifecycle (n.d.) and Snyk Open Source (n.d.) do include expert validation, their internal logic for metadata correction (e.g., CVE-to-package mappings or license risk categorization) is typically proprietary and undocumented. Users cannot verify why a particular component was flagged or enriched in a given way.

Furthermore, most SCA tools do not expose intermediate curation stages or support reproducible pipelines. When license scans or vulnerability traces change between tool versions, users lack the metadata to explain those differences. This contrasts sharply with NeMo and QURATOR, where curation stages are traceable, parameterized, and empirically testable.

Finally, the evaluation of curation in SCA tools remains informal or absent. While NeMo and QURATOR score metadata quality using defined filters or semantic models, SCA tools are rarely assessed on how well they validate or enrich retrieved metadata. Instead, evaluations focus on downstream metrics like false positives in vulnerability alerts, without clarifying whether those false positives stemmed from flawed retrieval or uncurated metadata.

Lessons for Advancing SCA Tool Curation

NeMo Curator and QURATOR offer important lessons for improving metadata curation in SCA:

- **Modularity and Traceability:** Curation workflows should be decomposed into identifiable steps with logs and outputs that can be audited and reused. This would enable SCA users to understand not just what the tool found, but how it derived its findings.
- **Explicit Evaluation of Curation Quality:** SCA tools could adopt quality scoring mechanisms for metadata (e.g., confidence levels for license detection or patch accuracy), similar to the content validation in NeMo and semantic enrichment in QURATOR.

- **Versioning and Reproducibility:** By version-controlling the curation pipeline and its configuration, SCA tools could make it possible to reproduce results across tool versions and track when and why metadata interpretations change.
- **Open Contribution Models:** Community-driven refinement, as seen in QURATOR, could supplement commercial curation by distributing the cost of expert validation and aligning updates with evolving standards.

In short, while SCA tools curation remains largely opaque and difficult to evaluate, models like NeMo Curator and QURATOR illustrate how structured, transparent, and evaluable pipelines can enhance trust, reproducibility, and collaborative improvement. Applying these design principles to SCA tools could significantly strengthen the quality and credibility of software metadata curation.

5.5 Observed Trends and Implications

This section synthesizes findings from a total of 31 unique SCA solutions, comprising 29 academic tools, including 20 standalone tools and 9 supporting components, as well as 2 additional tools identified through grey literature analysis (Sonatype Nexus Lifecycle and JFrog Xray). As previously noted in Section 4.5, each tool is counted only once, even if discussed in both academic and commercial sources, to ensure consistency and prevent duplication in the overall analysis.

The synthesis focuses on tool functionality (vulnerability- vs. license-focused), data source selection, and update frequency. These dimensions reveal important distinctions in how tools operate, what kinds of metadata they rely on, and how quickly they respond to ecosystem changes.

Emphasis on Vulnerability Detection Across SCA Solutions

Academic and commercial SCA research demonstrates a strong bias toward tools designed for vulnerability detection. Of the 20 standalone tools examined from academic literature, 16 (80%) were vulnerability-focused, while only 4 (20%) concentrated on license compliance. In contrast, among the 9 supporting components, 6 (67%) were license-focused and only 3 (33%) targeted vulnerabilities. This reversal suggests that license analysis is often handled by specialized components rather than full-fledged systems. When considering all 29 tools from academic sources, the overall trend still leans heavily toward vulnerability analysis, as detailed in Figure 4.4.

In comparison, six tools were identified from grey literature. Of these, four are commercial platforms, Sonatype Nexus Lifecycle, JFrog Xray, Snyk Open Source, and Black Duck. All four commercial tools are exclusively vulnerability-focused,

reinforcing the commercial sector's prioritization of vulnerability detection over license compliance.

Data Source Fragmentation Across Tool Types

A cross-comparison of primary data sources (visualized in Figure 4.5) reveals distinct trends by tool origin and focus. Academic tools heavily rely on standardized public datasets such as the NVD (n.d.), used by 8 tools, and SPDX (n.d.), used by 5 tools, favoring transparency and interoperability. In contrast, 9 tools, primarily commercial solutions, employ proprietary metadata feeds as their main data source. These vendor-maintained sources include enriched vulnerability databases, license catalogs, and internal threat intelligence, offering depth at the cost of transparency.

Tools categorized under "Other" sources (7 in total) frequently appear in academic literature and include handcrafted rule sets, license classifiers, and dataset-specific heuristics e.g., DIKE's license conflict matrix (Cui et al., 2023) or Ninka's regex-based engine (Higashi et al., 2023). While effective for narrow tasks, these custom sources can hinder interoperability and replicability due to lack of standardization.

Diverging Update Frequencies by Domain and Function

Update frequency is a critical factor in the practical relevance of SCA tools. As summarized in Table 2 (Appendix B), academic tools more often use static datasets or require manual updates. For example, VERJava (Sun et al., 2022) relies on a fixed benchmark snapshot, and LiDetector (Xu et al., 2023) offers no automatic sync with upstream feeds. This limits responsiveness to newly disclosed vulnerabilities or evolving license definitions.

In contrast, all four commercial tools analyzed, implement continuous update mechanisms. These tools ingest data from sources like the NVD, GHSA, and OSV, and enrich it with proprietary validation before synchronizing outputs with production environments. This enhances their timeliness but limits reproducibility due to proprietary logic and undocumented enrichment criteria.

Underreporting of Curation in Academic Evaluations

Despite the centrality of metadata curation to SCA accuracy, academic literature rarely documents how metadata is validated, corrected, or enriched. Among the academic tools and components analyzed, only a minority e.g., VULAS (Ponta et al., 2018), PatchScout (Tan et al., 2021), LiDetector (Xu et al., 2023), describe post-retrieval refinement steps, and even then, such steps are often treated as implicit. This lack of documentation hampers reproducibility and makes it difficult

to assess how much performance stems from detection accuracy versus curation quality.

Emphasis on Automation, with Opaque Workflows

Machine learning–based and rule-driven automation dominate both academic and commercial tools. Academic tools such as Holmes (Wu et al., 2024), Movery (Woo et al., 2022), and V1SCAN (Woo et al., 2023) use semantic code analysis, neural models, or static feature extraction to automate detection. Commercial tools also automate most scanning tasks, but unlike academic systems, they often combine this with structured manual intervention.

However, the internal workings of these pipelines are seldom made public. Model training data, enrichment rules, and validation logic remain proprietary, complicating independent verification and diminishing trust, especially in compliance-critical settings.

Community-Driven Platforms as Transparent Alternatives

Unlike commercial tools, community-led platforms such as ClearlyDefined (n.d.) and FOSSology (n.d.) offer public, transparent curation workflows. ClearlyDefined, in particular, supports GitHub-based contributions that record license metadata corrections and enhancements through pull requests, subject to community review (ClearlyDefined Curation, n.d.). Similarly, FOSSology integrates rule-based detection with manual override mechanisms and supports SPDX-compliant export for interoperability (FOSSology Workflow, n.d.). These platforms demonstrate that scalable, open, and reproducible curation is feasible when responsibilities are distributed across contributors rather than centralized within vendor teams.

Implications for Curation Transparency and Trust

The convergence of automation and manual validation in commercial tools has improved metadata quality but at the cost of transparency. Academic tools, while more interpretable, lack the update mechanisms and operational maturity needed for real-time application. Community-driven approaches strike a balance by enabling reproducibility and openness, though they require sustained participation.

Taken together, these findings suggest that future SCA tools should adopt modular, transparent curation pipelines. Such pipelines would separate detection, validation, and enrichment steps, document their logic, and expose results in standard formats. Moreover, integrating community contributions, through mechan-

isms like ClearlyDefined's Git-based review system, could distribute validation efforts while increasing accountability.

5.6 Limitations

This thesis presents a structured analysis of metadata curation in SCA tools. However, several limitations may affect the generalizability and robustness of its findings. These are discussed using the trustworthiness criteria of credibility, transferability, dependability, and confirmability, as outlined by Lincoln and Guba (1985).

5.6.1 Terminology Gaps and Interpretive Inclusion (Credibility)

A major challenge was the inconsistent or absent use of the term "curation" in the academic literature. Although this thesis defines curation as the validation, correction, and updating of license and vulnerability metadata, many papers did not frame their contributions in these terms. As a result, curation-related behaviors were often inferred from indirect evidence, such as dataset construction, scanner outputs, or update descriptions. While this interpretive inclusion made it possible to incorporate thematically relevant studies, it also introduces subjectivity that may influence the credibility of the analysis.

5.6.2 Limited Scope of Tool Types (Transferability)

The findings are based on 31 SCA solutions identified from academic and grey literature, including 22 standalone tools and 9 supporting components such as license classifiers or patch linkers. While these components offer detailed insight into specific curation tasks, they are not complete SCA systems and often operate in narrower or more controlled settings. As such, their workflows may not reflect the complexity of real-world, integrated toolchains. The dataset also includes 4 commercial platforms and 1 community-driven initiative, which helped incorporate practical perspectives. However, tools without public documentation or not discussed in the literature, particularly proprietary or enterprise-level systems, remain outside the scope of this study. Therefore, the results may not fully transfer to closed-source environments with undocumented curation processes.

5.6.3 Reliance on Documentation and Inference (Dependability)

In both academic and grey sources, curation workflows were often not described in full detail. Instead, this thesis had to reconstruct likely processes based on

available documentation, blog posts, or summarized tool features. This introduces a limitation to dependability: future research may interpret the same tools differently if vendor pages change, papers are revised, or unpublished processes become accessible. For example, while tools like Snyk Open Source (n.d.) and Black Duck (n.d.) outline aspects of curation in blogs and help articles, these sources are not always stable, complete, or technically granular.

5.6.4 Potential Bias in Literature Interpretation (Confirmability)

Both academic and grey literature present potential biases that affect confirmability. Academic papers may underreport practical aspects of curation, focusing instead on detection accuracy or conceptual models. Grey literature, especially vendor-authored material, tends to emphasize strengths while downplaying limitations. This disparity required triangulation across multiple sources to validate claims. Still, without independent benchmarks or third-party audits, the degree to which these findings can be independently confirmed remains limited. Interpretive bias may also be present due to the need to extrapolate curation details from partial or selectively presented information.

Summary Taken together, the findings highlight important gaps in the transparency and evaluation of metadata curation practices across both academic and commercial SCA tools. By comparing these practices with models from other domains, this chapter has shown how the field could benefit from more open, modular, and verifiable approaches to curation. The next chapter concludes the thesis and outlines potential directions for future research and tool development.

6 Conclusion and Future Work

6.1 Conclusion

This thesis examined a critical yet often overlooked aspect of SCA tools: the curation of metadata associated with open-source components. As outlined in the initial motivation, the increasing reliance on OSS has introduced significant security and compliance risks, risks that are often amplified by incomplete, outdated, or poorly curated metadata. To mitigate these risks, SCA tools have become essential, but their internal metadata validation and correction processes remain opaque and inconsistently reported. This study addressed this gap by systematically investigating how such metadata is curated, whether through manual, automated, or hybrid approaches.

A total of 31 unique SCA-related solutions were examined across academic literature and grey sources. This included a mix of standalone tools and supporting components, with both commercial and community-driven platforms contributing to the broader understanding of metadata curation. The inclusion of collaborative initiatives such as ClearlyDefined also offered valuable insight into transparent, community-based curation practices beyond traditional SCA tools.

The first research question explored current curation practices in SCA tools. This was addressed by categorizing tools or components into manual, automated, and hybrid types. In academic literature, vulnerability-focused tools primarily relied on automated curation, reflecting the emphasis on scalable detection techniques and static benchmarks. In contrast, commercial tools addressing vulnerabilities typically employed hybrid strategies, integrating automation with expert oversight. License-focused tools, especially in academic sources, relied more on manual or hybrid methods, reflecting the complexity of interpreting legal metadata.

The second research question asked how curation is described across academic and grey literature. Academic works were found to reference curation only implicitly, using terms like validation or correction. In contrast, commercial tools described curation more explicitly, but often without verifiable implementation

details. This highlighted a transparency gap: academic sources focus on technical evaluation, while commercial tools emphasize operational workflows, often without providing reproducible evidence.

The third research question addressed how tools are evaluated. The findings showed that most academic evaluations emphasize detection accuracy (e.g., precision, recall) without assessing the quality of metadata refinement. Grey literature focuses more on user-facing outcomes like reduced triage effort but lacks standardized metrics or independent validation. As a result, curation effectiveness remains an under-evaluated dimension in both domains.

The final research question asked whether insights from other fields could help improve SCA curation. Models from adjacent domains, including NeMo Curator by NVIDIA (n.d.) and QURATOR by Rehm et al. (2020), were shown to offer reproducible, modular, and auditable metadata workflows. These approaches provide a valuable blueprint for improving traceability and trust in metadata handling.

Methodologically, the study faced challenges due to inconsistent terminology, limited transparency, and the lack of formal documentation for many tools. These were addressed through an interpretive inclusion strategy, enabling the identification of relevant tools based on observable curation activities rather than terminology alone. Triangulation across documentation, blog posts, and academic descriptions helped ensure analytic rigor and reduce subjectivity.

In summary, the thesis achieved its primary objective: to expose and analyze the metadata curation practices in SCA tools. While the findings revealed significant gaps, particularly in transparency and evaluation, this work lays a foundation for more accountable and user-centered metadata management in future SCA tools. It also emphasizes the urgent need for clearer reporting standards and reproducible methodologies that allow both researchers and practitioners to assess curation quality with greater confidence.

6.2 Future Work

This study revealed several limitations that open important avenues for future research and tool development. A central challenge was the limited visibility into how metadata curation is performed within SCA tools. In both academic and grey literature, curation practices, such as validation, correction, and enrichment, are often only implied or fragmented across scattered documentation. In academic sources, the term ”curation” is rarely used at all, requiring interpretive analysis based on indirect evidence like rule sets, dataset construction, or error-handling behaviors. In commercial contexts, curation strategies are typically described in high-level terms or promotional narratives, lacking the transparency

or reproducibility needed to systematically evaluate tool behavior. This lack of clarity prevents users and researchers from understanding how metadata issues are handled, issues that directly impact the accuracy of vulnerability detection and license compliance.

To address these gaps, future research should focus on developing standardized benchmarks for evaluating metadata curation, not just scanner accuracy. This includes measurable indicators such as metadata correction rates, update traceability, and the ability to handle ambiguous or conflicting data. Standardizing these benchmarks would help compare tools more fairly and highlight meaningful differences in how metadata is maintained over time.

In addition, more user-centered research is needed to understand how developers interact with curated metadata. Inaccurate or unexplainable outputs reduce trust and increase remediation workload. By studying real-world use cases, future work can help design more transparent and interpretable workflows that clearly communicate when and how metadata has been verified or modified.

Future tools should also incorporate structured, auditable curation pipelines, allowing users to trace the origin and evolution of metadata. This can be inspired by models from other domains such as NeMo Curator by NVIDIA (n.d.), which demonstrates how versioning, modular design, and semantic enrichment can be implemented at scale. Community-driven platforms like ClearlyDefined (n.d.) further show that transparency and collaboration are achievable without sacrificing performance or scalability. Encouraging open contributions and expert feedback could reduce vendor lock-in and allow the broader software ecosystem to participate in improving metadata quality.

The groundwork laid in this thesis shows that while curation is under-discussed, it is central to the effectiveness of SCA tools. The study developed a typology of curation methods and surfaced patterns in how tools approach metadata refinement. These insights provide a foundation for advancing both research and practice in this space. By extending this work into future projects, whether through evaluation frameworks, open datasets, or participatory platforms, the community can build SCA tools that are not only functionally capable but also trustworthy, explainable, and resilient in the face of evolving software supply chain risks.

Appendices

A Primary Data Sources for Analyzed SCA Solutions

Table 1: Primary Data Sources for Academic and Commercial SCA Solutions

Tool/Component	**Primary Data Source**
LiDetector	SPDX License List
DIKE	Internal license knowledge base
ScanCode	SPDX License List
FOSSology	SPDX License List
V1SCAN	NVD
ODC	NVD
GitHub Dependabot	GHSA
Grype	Anchore's curated database (mirrors NVD, GHSA, etc.)
OSV Scanner	OSV database
Snyk Open Source	Proprietary database (GitHub Advisories, NVD)
VULAS	Proprietary database (SAP-internal vulnerability database)
Black Duck	Proprietary DB
Cybellum	Proprietary DB
Scantist	Proprietary DB
VODA	NVD
Vul4Java	NVD
OSS Index	Public sources
PatchScout	NVD
VERJava	NVD
Movery	NVD
Ninka	Manually curated license rules
Holmes	NVD
findOSSLicense	Compatibility matrix, Libraries.io, scanner rules
FOSSLT	SPDX License List
OMP	Proprietary (Maven repositories)
ASLA	Custom DB (Manually curated regex-based)
Eclipse Steady	Kaybee project
npm audit	Proprietary (npm Security Advisory Database)
GitHub Licensee	SPDX License List
JFrog Xray	Proprietary
Sonatype Nexus Lifecycle	Proprietary

B Update Frequency for Academic and Commercial SCA Solutions

Table 2: Update Frequency for Academic and Commercial SCA Solutions

Tool or Component	Update Frequency Type
LiDetector	Static
DIKE	Manual
ScanCode	Manual
FOSSology	Manual
V1SCAN	Static
ODC	Continuous
GitHub Dependabot	Continuous
Grype	Continuous
OSV Scanner	Continuous
Snyk Open Source	Continuous
VULAS	Manual
Eclipse Steady	Continuous
Black Duck	Continuous
Cybellum	Continuous
Scantist	Continuous
VODA	Continuous
Vul4Java	Static
OSS Index	Continuous
PatchScout	Static
VERJava	Static
Movery	Static
Ninka	Manual
Holmes	Static
findOSSLicense	Manual
FOSSLT	Static
OMP	Manual
ASLA	Manual
npm audit	Continuous
GitHub Licensee	Manual
JFrog Xray	Continuous
Sonatype Nexus Lifecycle	Continuous

C Codebook for Tool Analysis Variables

This codebook defines the core variables used to analyze SCA tools and components in this study. Each variable is accompanied by a brief description and the categories or values used during data extraction and synthesis.

Tool or Component Name

Identifies the specific SCA solution under examination. This may be a standalone tool (e.g., Black Duck, Snyk Open Source) or a modular component (e.g., Ninka) that supports broader SCA workflows.

Primary Focus

Specifies whether the tool primarily targets vulnerability detection or license compliance, based on its stated design purpose.

- **Vulnerability:** Focuses on identifying, classifying, and mitigating software vulnerabilities.
- **License:** Aims to detect, validate, and manage software license metadata and compliance.

Curation Methodology

This category captures how the tool refines or improves retrieved metadata after its initial collection. As described in Section 1.2, curation includes processes such as validation, correction, updating, and enrichment of metadata.

Each study was analyzed according to this definition to assess whether and how the tool it describes performs metadata curation.

Curation tasks are categorized as follows:

- **Validation:** Verifying the accuracy, consistency, or completeness of metadata.
- **Correction:** Fixing errors or inaccuracies in the collected metadata.
- **Updating:** Replacing outdated or obsolete metadata with more current information.
- **Enrichment:** Adding missing context, details, or enhancements to metadata to improve its quality and utility.

Based on how these tasks are performed, the curation methodology is classified into:

- **Manual:** All curation activities are performed by humans, typically involving expert review or manual editing.
- **Automatic:** Curation is fully automated using predefined rules, heuristics, or machine learning techniques.
- **Hybrid:** A combination of automatic methods and human oversight or intervention, where automation handles routine cases and humans address edge cases or quality assurance.

Primary Data Source

In the context of SCA tools, a **data source** refers to an external repository or database that a tool relies on to identify, validate, or enhance metadata about software components, particularly licenses and known vulnerabilities.

Common Data Sources in SCA Tools

- **NVD:** National Vulnerability Database.
- **SPDX:** Software Package Data Exchange license list.
- **OSS Index:** Vulnerability data service by Sonatype.
- **Proprietary:** Closed, vendor-maintained or internal databases.

Update Frequency

Indicates how frequently the tool refreshes its underlying metadata.

- **Manual:** Updates are user-triggered or infrequent.
- **Static:** Relies on pre-packaged datasets updated occasionally.
- **Continuous:** Updates are automated and frequent, often via real-time synchronization with external APIs or feeds.

References

Bhandari, G., Naseer, A., & Moonen, L. (2021). Cvefixes: Automated collection of vulnerabilities and their fixes from open-source software. *Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '21)*, 95–104. https://doi.org/10.1145/3475960.3475985

Black Duck. (n.d.). Black duck software composition analysis documentation [Accessed April 2025]. https://documentation.blackduck.com/bundle/bd-hub-2024.10/page/Risk/VulnImpact.html

Black Duck. (2022). Black duck: Black duck blog post [Accessed April 2025]. https://www.blackduck.com/blog/comparing-bdsa-with-nvd-version-accuracy.html

Black Duck KnowledgeBase. (n.d.). Black duck:black duck knowledgebase [Accessed April 2025]. https://www.blackduck.com/software-composition-analysis-tools/knowledgebase.html

Bottner, L., Hermann, A., Eppler, J., Thüm, T., & Kargl, F. (2023). Evaluation of free and open source tools for automated software composition analysis. *Proceedings of the 7th ACM Computer Science in Cars Symposium (CSCS)*, 3:1–3:11. https://doi.org/10.1145/3631204.3631862

Chen, Y., Santosa, A. E., Yi, A. M., Sharma, A., Sharma, A., & Lo, D. (2020). A machine learning approach for vulnerability curation. *Proceedings of the 17th International Conference on Mining Software Repositories*, 32–42. https://doi.org/10.1145/3379597.3387461

ClearlyDefined. (n.d.). Clearlydefined: Clearlydefined documentation [Accessed April 2025]. https://docs.clearlydefined.io/docs/get-involved/intro

ClearlyDefined Curation. (n.d.). Clearlydefined: Curation and upstreaming [Accessed April 2025]. https://docs.clearlydefined.io/docs/get-involved/data-curation

Cui, X., Wu, J., Wu, Y., Wang, X., Luo, T., Qu, S., Ling, X., & Yang, M. (2023). An empirical study of license conflict in free and open source software. *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 495–505. https://doi.org/10.1109/ICSE-SEIP58684.2023.00050

Dann, A., Plate, H., Hermann, B., Ponta, S., & Bodden, E. (2022). Identifying challenges for oss vulnerability scanners - a study test suite. *IEEE Transactions on Software Engineering*, *48*, 3613–3625. https://doi.org/10.1109/TSE.2021.3101739

Danny Allan. (2025). Snyk: Snyk blog post [Accessed April 2025]. https://snyk.io/blog/snyks-statement-on-the-mitre-cves-program-funding-update/

Dietrich, J., Pfoh, T., & Bodden, E. (2023). Hidden in plain sight: Undetected vulnerable clones in java projects. *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 1–12. https://arxiv.org/abs/2306.05534

Dyck, S., Haferkorn, D., Kerth, C., & Schoebel, A. (2018). Automating open source software license information generation in software projects. *Systemics, Cybernetics and Informatics*, *16*(5), 44–49. https://www.iiisci.org/journal/CV$/sci/pdfs/SA117XZ18.pdf

FOSSology. (n.d.). Fossology: Fossology features section [Accessed April 2025]. https://www.fossology.org/features/

FOSSology Workflow. (n.d.). Fossology: Fossology basic workflow [Accessed April 2025]. https://www.fossology.org/get-started/basic-workflow/

GHSA. (n.d.). Github security advisories [Accessed: 2025-04-25]. https://github.com/advisories

Higashi, Y., Ohira, M., & Manabe, Y. (2023). Automating license rule generation to help maintain rule-based oss license identification tools. *Journal of Information Processing*, *31*, 2–12. https://doi.org/10.2197/ipsjjip.31.2

Imtiaz, N., Thorn, S., & Williams, L. (2021). A comparative study of vulnerability reporting by software composition analysis tools. *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. https://doi.org/10.1145/3475716.3475769

Ivanova, E., Stakhanova, N., & Sistany, B. (2024). Adversarial analysis of software composition analysis tools. *Proceedings of the 27th Information Security Conference (ISC 2024)*, 161–182. https://doi.org/10.1007/978-3-031-75764-8_9

Jeff McAffer. (2019). Clearlydefined: Crowdsourcing license compliance with clearlydefined [Accessed April 2025]. https://opensource.com/article/19/5/license-compliance-clearlydefined

Jennings, J., Patwary, M., Subramanian, S., Prabhumoye, S., Dattagupta, A., Shoeybi, M., & Catanzaro, B. (2023). *Curating trillion-token datasets: Introducing nemo data curator* [Accessed April 2025]. https://developer.nvidia.com/blog/curating-trillion-token-datasets-introducing-nemo-data-curator/

JFrog. (n.d.). Jfrog security cve research and enrichment [Accessed April 2025]. https://jfrog.com/curation/

JFrog Xray. (n.d.). Jfrog security cve research and enrichment [Accessed April 2025]. https://jfrog.com/help/r/jfrog-security-user-guide/products/xray

JFrog Xray License. (n.d.). Jfrog: Jfrog xray documentation. https://jfrog.com/help/r/jfrog-security-user-guide/products/xray/features-and-capabilities/sca/legal

Kapitsaki, G. M., Paphitou, A. C., & Achilleos, A. (2022). Towards open source software licenses compatibility check. *Proceedings of the 26th Pan-Hellenic Conference on Informatics (PCI 2022)*, 96–101. https://doi.org/10.1145/3575879.3575973

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, *33*.

Kumar, S. H. B. I., Sampaio, L. R., Martin, A., Brito, A., & Fetzer, C. (2024). A comprehensive study on the impact of vulnerable dependencies on open-source software. *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*, 96–107. https://doi.org/10.1109/ISSRE62328.2024.00020

Lincoln, Y. ., & Guba, E. G. (1985). *Naturalistic inquiry*. Sage Publications.

Nguyen, B. (2024). *Processing high-quality vietnamese language data with nvidia nemo curator* [Accessed April 2025]. https://developer.nvidia.com/blog/processing-high-quality-vietnamese-language-data-with-nvidia-nemo-curator/

Nick Vidal. (2024). Clearlydefined: Open source initiative [Accessed April 2025]. https://opensource.org/blog/clearlydefined-at-soss-fusion-2024-a-collaborative-solution-to-open-source-license-compliance

Ning, Y., Zhang, Y., Ma, C., Guo, Z., & Yu, L. (2024). Empirical study of software composition analysis tools for c/c++ binary programs. *IEEE Access*, *12*, 50418–50430. https://doi.org/10.1109/ACCESS.2023.3341224

NVD. (n.d.). National vulnerability database (nvd) [Accessed: 2025-04-25]. https://nvd.nist.gov/vuln

NVIDIA. (n.d.). Nemo curator: Overview [Accessed April 2025]. https://developer.nvidia.com/nemo-curator

NVIDIA GitHub documentation. (n.d.). Nemo curator: Modular and reproducible dataset curation pipeline [Accessed April 2025]. https://github.com/NVIDIA/NeMo-Curator

Ombredanne, P. (2020). Free and open source software license compliance: Tools for software composition analysis. *Computer*, *53*(10), 105–109. https://doi.org/10.1109/MC.2020.3011082

OSSRA. (2025). 2025 open source security and risk analysis (ossra) report [Accessed April 2025]. https://www.blackduck.com/blog/open-source-trends-ossra-report.html

OSV. (n.d.). Osv vulnerabilities [Accessed: 2025-04-25]. https://osv.dev/list

Ouzan, O., & Shalom, J. S. (2025). Jfrog: Jfrog blog post [Accessed April 2025]. https://jfrog.com/blog/mitres-close-call-in-cve-management/

Ponta, S. E., Plate, H., Sabetta, A., Bezzi, M., & Dangremont, C. (2019). A manually-curated dataset of fixes to vulnerabilities of open-source soft-

ware. *Proceedings of the 16th International Conference on Mining Software Repositories (MSR)*, 383–387. https://doi.org/10.1109/MSR.2019.00064

Ponta, S. E., Plate, H., & Sabetta, A. (2018). Beyond metadata: Code-centric and usage-based analysis of known vulnerabilities in open-source software. *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 449–460. https://doi.org/10.1109/ICSME.2018.00054

Rehm, G., Bourgonje, P., Hegele, S., Kintzel, F., Schneider, J. M., Ostendorff, M., Zaczynska, K., Berger, A., Grill, S., Räuchle, S., Rauenbusch, J., Rutenburg, L., Schmidt, A., Wild, M., Hoffmann, H., Fink, J., Schulz, S., Seva, J., Quantz, J., … Heine, F. (2020). Qurator: Innovative technologies for content and data curation. https://arxiv.org/abs/2004.12195

ScanCode. (n.d.). Find open source with open source, with scancode. [Accessed: 2025-04-25]. https://aboutcode.org/scancode/

Security, O. (2024). Everything you need to know about software composition analysis (sca) [Accessed April 2025]. https://www.ox.security/everything-you-need-to-know-about-software-composition-analysis-sca/

Sharma, P., Shi, Z., Şimşek, Ş., Starobinski, D., & Medina, D. (2024). Understanding similarities and differences between software composition analysis tools. *IEEE Security Privacy*, *PP*, 2–13. https://doi.org/10.1109/MSEC.2024.3410957

Snyk & Foundation, T. L. (2024, March). The State of Open Source Security 2024. https://snyk.io/blog/2024-open-source-security-report-slowing-progress-and-new-challenges-for/

Snyk Curation. (n.d.). Snyk: Snyk's documentation [Accessed April 2025]. https://github.com/snyk/user-docs/blob/main/docs/manage-risk/prioritize-issues-for-fixing/malicious-packages.md

Snyk Open Source. (n.d.). Snyk open source documentation [Accessed April 2025]. https://docs.snyk.io/scan-with-snyk/snyk-open-source

Snyk User Docs. (n.d.). Snyk: Snyk's license documentation [Accessed April 2025]. https://docs.snyk.io/scan-with-snyk/snyk-open-source/scan-open-source-libraries-and-licenses/

Sonatype License. (n.d.). Sonatype: "component license information" [Accessed April 2025]. https://help.sonatype.com/en/component-license-information.html

Sonatype Nexus Lifecycle. (n.d.). *Understand how risks to projects from open source software can be managed through software composition analysis.* [Accessed April 2025]. https://www.sonatype.com/resources/articles/what-is-software-composition-analysis

Sonatype Vulnerability. (n.d.). Sonatype: "sonatype vulnerability data" [Accessed April 2025]. https://help.sonatype.com/en/sonatype-vulnerability-data.html

SPDX. (n.d.). Spdx: Software package data exchange [Accessed: 2025-04-25]. https://spdx.dev

Sun, Q., Xu, L., Xiao, Y., Li, F., Su, H., Liu, Y., Huang, H., & Huo, W. (2022). Verjava: Vulnerable version identification for java oss with a two-stage analysis. *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 329–339. https://doi.org/10.1109/ICSME55016.2022.00037

Synopsys. (2024). *Override severity - black duck user guide* [Accessed: 2025-04-22]. https://sig-product-docs.synopsys.com/bundle/srm/page/user_guide/Findings/override_severity.html

Tan, X., Zhang, Y., Mi, C., Cao, J., Sun, K., Lin, Y., & Yang, M. (2021). Locating the security patches for disclosed oss vulnerabilities with vulnerability-commit correlation ranking. *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021)*, 952–964. https://doi.org/10.1145/3460120.3484593

Tuunanen, T., Koskinen, J., & Kärkkäinen, T. (2006). Retrieving open source software licenses. *Open Source Systems*, 35–46. https://doi.org/10.1007/0-387-34226-5_4

Wang, Z., Hu, J., Zhou, Y., Tambadou, S., & Zuo, F. (2024). Vul4java: A java oss vulnerability identification method based on a two-stage analysis. *Proceedings of the International Conference on Algorithms, Software Engineering, and Network Security (ASENS 2024)*, 742–746. https://doi.org/10.1145/3677182.3677315

Woo, S., Choi, E., Lee, H., & Oh, H. (2023). V1SCAN: Discovering 1-day vulnerabilities in reused C/C++ open-source software components using code classification techniques. *Proceedings of the 32nd USENIX Security Symposium*, 6541–6552. https://www.usenix.org/conference/usenixsecurity23/presentation/woo

Woo, S., Hong, H., Choi, E., & Lee, H. (2022). MOVERY: A precise approach for modified vulnerable code clone discovery from modified Open-Source software components. *31st USENIX Security Symposium (USENIX Security 22)*, 3037–3053. https://www.usenix.org/conference/usenixsecurity22/presentation/woo

Wu, S., Song, W., Huang, K., Chen, B., & Peng, X. (2024). Identifying affected libraries and their ecosystems for open source software vulnerabilities. *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. https://doi.org/10.1145/3597503.3639582

Xu, S., Gao, Y., Fan, L., Liu, Z., Liu, Y., & Ji, H. (2023). Lidetector: License incompatibility detection for open source software. *arXiv preprint arXiv:2204.10502*. https://arxiv.org/abs/2204.10502

Zhao, J., Ren, Y., Li, B., Zhao, X., & Zhang, J. (2023). Fosslt: An efficient model for automatic finding open source software license texts. *2023 2nd International Conference on Cloud Computing, Big Data Application and Software*

Engineering (CBASE), 269–275. https://doi.org/10.1109/CBASE60015.2023.10439089

Zhao, L., Chen, S., Xu, Z., Liu, C., Zhang, L., Wu, J., Sun, J., & Liu, Y. (2023). Software composition analysis for vulnerability detection: An empirical study on java projects. *Proceedings of the 31st ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023)*, 960–972. https://doi.org/10.1145/3611643.3616299