The International Journal of Public Policies In Egypt- Volume 3, Issue 4 (October 2024)
ISSN: Print: 2812-4758, Online: 2812- 4766
Published by IDSC

IJPPE

Comparative Evaluation of Software Composition Analysis Tools in Context of Technical Debt Reduction

Nada M. Ali[1], Samar K. Mohamed[2], Dalia A. Magdi[3]

[1] School of Computer Science, Canadian International College, Cairo, Egypt

[2] School of Computer Science, Canadian International College, Cairo, Egypt

[3] Computer and Information System Department, School of Computer Sciences, Sadat Academy for Management Sciences, Cairo, Egypt

Correspondence: Daliamagdi@gmail.com

التقييم المقارن لأدوات تحليل تكوين البرمجيات في سياق تخفيض الديون الفنية

ندا مصطفى علي [1]، سمر خالد محمد [2]، داليا أحمد مجدي [3]

[1] كلية علوم الحاسب، المعهد الكندي العالي للحاسب الألى، القاهرة، مصر

[2] كلية علوم الحاسب، المعهد الكندي العالي للحاسب الألى، القاهرة، مصر

[3] قسم نظم المعلومات والحاسب الآلي، كلية علوم الحاسب الآلي، أكاديمية السادات للعلوم الإدارية، القاهرة، مصر

المراسلة: Daliamagdi@gmail.com

- **DOI: 10.21608/ijppe.2024.389252** **URL: http://doi.org/ 10.21608/ijppe.2024.389252**
- Received: 13/6/2024, Accepted: 25/9/2024
- Citation: Ali, N., Mohamed, S., & Magdi, D. (2024). Comparative Evaluation of Software Composition Analysis Tools in Context of Technical Debt Reduction. The International Journal of Public Policies in Egypt, 3(4), 206 - 228.

The International Journal of Public Policies In Egypt, Volume 3, Issue 3 (July 2024)
ISSN: Print: 2812-4758, Online: 2812- 4766
Published by IDSC

Comparative Evaluation of Software Composition Analysis Tools in Context of Technical Debt Reduction

Abstract

The metaphor of "technical debt" is used in software engineering to describe technical solutions that may be practical in the short term but have a detrimental long-term consequence. Tools for software composition analysis (SCA) are proposed to detect potential vulnerabilities presented by open-source software (OSS) imported as third-party libraries. As software functionality becomes more complicated, SCA tools may confront various scenarios throughout the dependency resolution process, including diverse artifact formats, dependency imports, and dependence requirements. This study provides a comparative review of SCA techniques in the context of technical debt reduction, focusing on the analogous decisions and dynamics seen in systems engineering.

***Keywords*:** Technical Debt, Software Composition Analysis (SCA), Open-Source Software (OSS), dependency resolution, systems engineering

Introduction

In software engineering, "technical debt" is commonly used to describe the trade-off between rapid development and the long-term maintainability of a software system (Dudee, 2021). This metaphor, popularized by Ward Cunningham (Melo et al., 2022), represents the potential consequences of prioritizing rapid development over strong codebases. A very dangerous type of technical debt is "security debt," which results from vulnerabilities introduced during the software development lifecycle. If not fixed, these security flaws might expose systems to attack, potentially resulting in serious consequences.

The widespread use of open-source software (OSS) has had a considerable impact on development processes, with developers increasingly turning to third-party libraries (TPLs) to speed up development and improve application functionality (Zaimi et al., 2015). While this technique reduces time-to-market, it also raises possible security issues, contributing to building security debt.

Software Composition Analysis (SCA) tools have evolved as critical components of current software development processes to address these concerns. These tools check software projects for vulnerabilities, license violations, and other security threats (Ombredanne, 2020). By recognizing these concerns early in the development process, SCA tools can assist organizations in reducing technical debt, particularly security debt, and improving software quality.

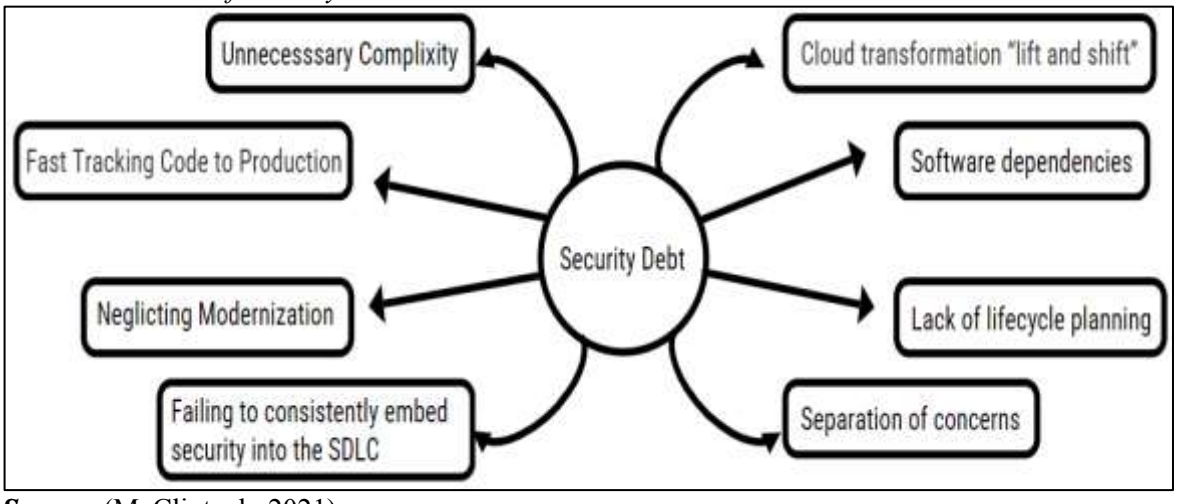
However, the value of SCA tools varies greatly, depending on aspects like the tool's accuracy, performance, and the complexity of the software under analysis. The purpose of this study is to evaluate SCA tools comprehensively to determine their usefulness in reducing security debt and to give actionable information to organizations looking for ways to improve their software development processes.

Security Debt

It is one of the technical debt types that refers to the vulnerabilities and bugs in a software system during its entire life cycle. It is the common result of decisions taken during the development process that value speed of delivery over effective controls (Siavvas et al., 2019). The following figure illustrates the various sources that contribute to security debt within a software system.

Figure 1

Common Sources of Security Debt



Source: (McClintock, 2021).

As shown in Figure 1, the sources of accruing security debt can be severe. Unlike other types of technical debt, which primarily influence maintainability and performance, security debt poses a significant threat to a software system's integrity and safety.

Managing security debt requires a proactive strategy for addressing vulnerabilities. Software Composition Analysis (SCA) tools are widely used for this purpose. By continuously monitoring and analyzing the open-source components of a software project, SCA tools can find vulnerabilities early and provide actionable insights to mitigate them. The proper use of SCA tools can dramatically decrease security debt, ensuring that software is secure and resilient to possible threats.

The major objective of this research is to compare software composition analysis tools in the context of technical debt reduction. The first step toward addressing this objective is to evaluate various software composition analysis tools and analyze the results and findings. These tools are then compared and evaluated based on seven different criteria.

After the introduction, this study is divided into sections: section 1 discusses the literature review, section 2 tackles the importance of technical debt reductions, section 3 discusses the selection of the most appropriate SCA tools, and the last section is the conclusion and future directions.

Literature Review

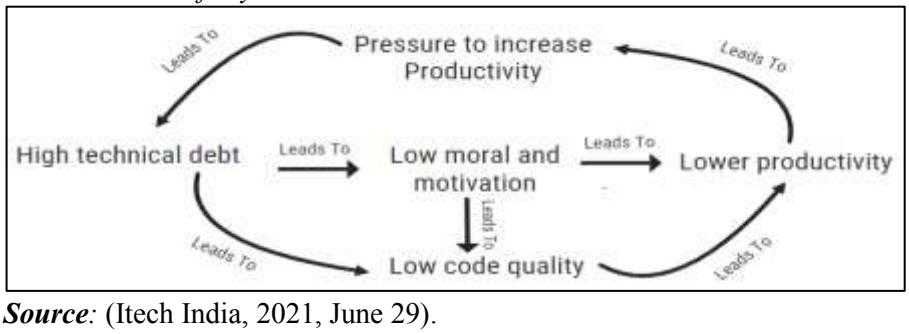
This section critically examines existing research on technical debt, aiming at identifying key findings, theories, and methodologies that can guide future studies. By analyzing current work, this evaluation will also pinpoint gaps in our understanding of technical debt and explain how the current study uniquely contributes to this field.

Definition and Concept of Technical Debt

Cunningham first introduced the technical debt metaphor to explain the importance of refactoring software to his management system (Melo et al., 2022). Technical debt describes the long-term consequences of implementation decisions made during the software development process. These implementation decisions concentrate on immediate benefits, such as shorter development time or feature delivery, over long-term considerations, resulting in lower code quality, higher complexity, and decreased maintainability. The technical debt life cycle describes how it may be introduced, managed, and ultimately resolved within a software development project. Figure 2 illustrates the life cycle of technical debt from its inception to its resolution, highlighting key stages and potential consequences.

Figure 2

Technical debt life cycle



Source: (Itech India, 2021, June 29).

Understanding the life cycle of technical debt, as shown in Figure 2, is essential for effectively controlling and minimizing its negative effects on software quality and sustainability. Different types

of technical debt were introduced, including design, code, test, documentation, and security debt. Each one influences a different aspect of software development (Li et al., 2023), as:

- Design debt refers to architectural or design decisions that consider short-term gains but may hinder future scalability or extensibility.
- Code debt occurs when developers choose an easy way to finish a feature instead of the best practice, which could take more time.
- Test debt happens when testing tasks are postponed or compromised during software development.
- Documentation debt is a lack of or outdated documentation that prevents the understanding and maintenance of a software system.
- Security debt focuses on vulnerabilities and weaknesses developed during the software development process, which provide security risks that attackers can exploit.

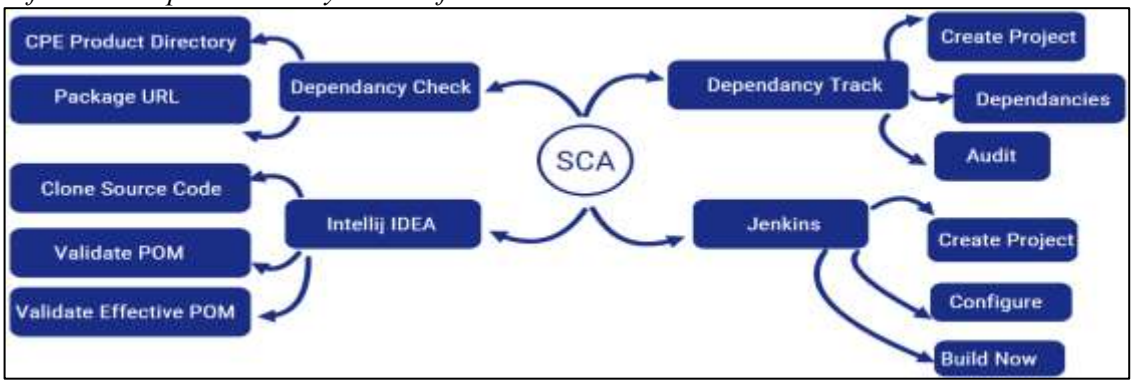
Importance of Composition Analysis Tools

Software composition analysis (SCA) is a technique used to discover and manage open-source components and licensing (Ombredanne, 2020). As shown in Figure 3, this technique uses a thorough analytic process to discover vulnerabilities. It ensures that all components, including open-source dependencies, meet demanding quality and security criteria before being integrated. This comprehensive strategy helps prevent threats while improving the software's overall security and dependability.

For any modern-day software product, the equivalent, or more, number of lines of code of open-source software is utilized. From Stack Overflow, GitHub, PyPi, or any code-related query online, you will find results and offer more reusable code that can be directly plugged into any product. SCA provides higher speed, convenience, better solutions, debugged versions, and smaller investments, which are helpful for start-ups or larger organizations (Imtiaz et al., 2021). Figure 3 illustrates the sequential steps involved in the software composition analysis process.

Figure 3

Software Composition Analysis Workflow



Source: (TatvaSoft Blog, 2023, December 12)

These tools play an important role in modern software development environments for several compelling reasons:

- Tracking open-source components, SCA tools help automatically detect the open-source libraries used and generate a report, ensuring developers know what they are using and can quickly address potential issues.
- Eliminating Business Risks, while open-source components might be advantageous, they can also pose unexpected threats to a firm. Using old or vulnerable components exposes software to attack. SCA assists in recognizing these risks early, allowing firms to take proactive measures to ensure software stability.

- Continuous vulnerability detection and monitoring, it is important to have a continuous monitoring technique because new vulnerabilities are identified daily. SCA composition analysis offers continuous vulnerability detection and monitoring, highlighting vulnerabilities in real-time and enabling quick action.
- Automated and Prioritized Vulnerability Remediation, one of the most notable aspects of SCA is its ability to not only discover vulnerabilities but also prioritize them depending on severity. This ensures that the most serious vulnerabilities are addressed first, automating most of the remediation process and making better use of resources.
- Reducing security costs, investing in (SCA) Software Composition Analysis may appear to be an additional expenditure, but in the long term, it considerably saves security costs. You can avoid costly patches and potential breaches by identifying vulnerabilities early on.

This research aims to fill the gaps by doing a comparative analysis of SCA tools, focusing on their ability to decrease security debt. It provides a complete review to help organizations choose the best SCA tool for their needs by assessing essential features, performance, and integration possibilities.

Related Works

To speed up development, software engineers usually create security debt, a backlog of security vulnerabilities that must be fixed later. This section examines existing research on recognizing, quantifying, and managing security debt to improve software system security. While previous research has improved our knowledge of security debt, important research gaps remain, as there are no clear criteria for evaluating security debt.

Understanding the consequences of ignoring security debt is critical since it can result in serious system vulnerabilities, financial losses, and reputational harm. This research will develop a uniform methodology for quantifying security debt and analyze the link between security debt and software quality attributes. By filling these research gaps, this study contributes to a better understanding of security debt and its effects on software systems.

Coetzer and Leenen (2024) delved into cybersecurity debt, mentioning that cybersecurity debt is a form of technical debt that focuses on finding security vulnerabilities in IT systems, which increases due to resource limitations, time constraints, and a lack of expertise, potentially leading to serious security breaches. The study highlighted the importance of identifying, prioritizing, and mitigating cyber security debt and the escalating risks of delaying its repayment. Using a detailed analysis and a case study of the Equifax breach, the study defined the real-world consequences of skipping security debt management.

Cifuentes et al. (2023) researched developing program analysis tools for finding security vulnerabilities in industrial environments. The study underlined that the successful utilization of these tools by development companies depends on low false-positive rates, ease of integration, scalability, and straightforward results. Analysis techniques have evolved to address a variety of programming languages and security concerns, including memory-related vulnerabilities in C and injection vulnerabilities in Javascript and Python. The study proposed an intelligent application security vision in which integrated technologies share information and address new concerns, such as supply chain security.

Kruke (2022) investigated the concept of security debt within software systems, defining it as choices that could be design choices or implementation choices. These choices could slow down achieving the optimal security goal. By doing an exploratory case study on 26 different software, the study dived into how security debt is managed and how it can be a part of technical debt. The study defined three main methods for managing security debt. The result indicated the necessity of security-

oriented management approaches and revealed that inadequate security knowledge can lead to increased security debt. The paper also settled a connection between security debt and technical debt.

Imtiaz et al. (2021) investigated the differences in vulnerability reporting using different software composition analysis tools, which are used to follow vulnerabilities in third-party libraries and frameworks. By analyzing nine industry-leading SCA tools on a large web application, OpenMRS, which includes Maven (Java) and npm (JavaScript) projects, this research demonstrated the differences in the number of vulnerabilities that have been reported using different tools, ranging from 17 to 332 for Maven and 32 to 239 for npm projects.

Martinez et al. (2021) demonstrated proactive security management in diverse industries. The paper focused on applying inadequate solutions to achieve desired security levels, highlighting the challenges companies face in addressing and explicitly stating security debt items and defining security debt as a result.

Rindell et al. (2019) focused on discussing the under-prioritization of security in software development, in which developers sometimes lack awareness of security practices. The study proposed identifying security risk as a type of technical debt. Based on this proposal, it identified the concept of security debt, which encompassed security risks within TD categories such as requirements, architecture, code, and testing.

Izurieta et al. (2018) addressed the management of technical debt in the context of security breaches identified through the design phase of software development. The study goal was to establish a method for finding TD linked to security weaknesses. This study also defined security debt as a special case within the technical debt management system that should be considered due to the potential different business impacts of unfounded security weaknesses.

Technical debt is a metaphor for the consequences of poor technical decisions, and it includes security debt, which is produced by ignoring security during software development. Viewing security debt through the perspective of technical debt theory allows us to identify underlying causes, prioritize essential efforts, and design effective management solutions. Existing review papers generally address the nature, effects, and management of security debt. Table 1 compares the current study with other review studies, highlighting definitions, quantification, and management strategies.

Table 1

Comparison Between Other Review Papers

Study	Definition of Security Debt	Quantification Method	Management Strategies
Coetzer& Leenen (2024)	Form of technical debt related to security vulnerabilities	Risk assessment	Prioritization and mitigation of cyber security debt
Kruke (2022)	Choices that limit achieving optimal security goals	Qualitative analysis	Security-oriented management approaches
Martinez et al. (2021)	Inadequate solutions to achieve desired security levels	Case studies	Proactive security management
Current Study	vulnerabilities and bugs that have happened in a software system during its entire life cycle	Key features and capabilities of each SCA tool	Vulnerability Management

Source: Prepared by the authors.

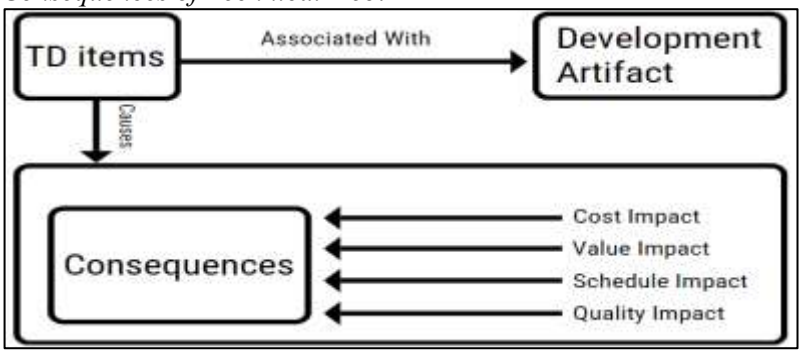
SCA tools, which look for vulnerabilities in software components, appear to be a good solution to manage security debt. While few studies have mentioned SCA, a lack of attention is paid to its significance in decreasing security debt. This study seeks to close this gap by thoroughly evaluating SCA tools and their usefulness in finding, prioritizing, and fixing vulnerabilities. By overcoming past research constraints and conducting a targeted investigation of SCA tools, it contributes to a better understanding of security debt. It provides practical insights for organizations looking to enhance their software security posture.

Importance of Technical Debt Reduction

According to Stepsize's survey (Stepsize, 2021), 58% of businesses lack a mechanism for reducing technical debt, despite 60% of engineers warning about the negative impact on the company. Figure 4 shows the negative consequences of increasing technical debt in software development. It highlights the possible consequences on project schedules, development costs, code quality, and overall system performance. Figure 4 illustrates the consequences of accumulated technical debt on software development projects.

Figure 4

Consequences of Technical Debt



Source: (Olsson et al., 2021).

Working with outdated systems becomes more difficult than it should be due to technical debt. It's an uncomfortable reality for developers and entrepreneurs. Moreover, the cost of technical debt rises considerably with time. Thus, the sooner your business addresses these loans in your codebases, the better.

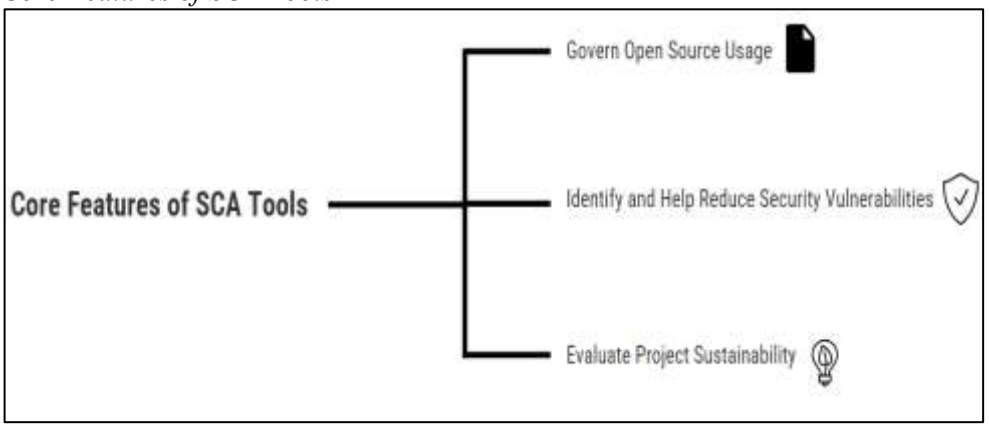
Figure 5 shows the significant advantages that can be achieved by effectively reducing technical debt. The figure outlines three primary categories of benefits, each with its own set of positive outcomes:

1. Productivity, reducing technical debt makes it easier for the development team to provide new features, resolve defects, and deliver high-quality software. It also improves their morale because they can provide more value faster.

2. Product quality, a well-planned system can support a wider feature set than a badly developed one. As a result, reducing technical debt can help create better products with fewer bugs and vulnerabilities.

3. Maintainability and scalability, a good codebase is much easier to maintain than an ill-conceived one. Reducing technical debt will reduce the cost of overall future maintenance because it will be much easier for the developers to understand the codebase and immediately start working on it.

Figure 5
Core Features of SCA Tools



***Source*:** (Debricked, 2021- October 11).

Reducing technical debt is an important factor for success and efficiency in modern software development, as unfixed technical debt leads to software errors. By managing and reducing technical debt wisely, businesses can enhance development processes and product reliability and make sure that their systems remain adaptable and robust in the face of future challenges (Rios et al., 2018).

Methodology

A combination of tools, including static code analysis, vulnerability scanners, and risk assessment frameworks, are essential for calculating security debt and setting priorities in software development projects.

Accurately measuring security debt and successfully prioritizing vulnerabilities are still important issues in software development. While several tools and methodologies have been developed, a standardized approach to these tasks remains absent. This study intends to address these problems by performing a comparative review of existing methods for assessing and analyzing security debt reduction measures. By studying these tools, we want to determine their strengths, shortcomings, and prospects for solving the challenges of security debt management.

This section thoroughly explains the methodologies used to assess and analyze various options for lowering security debt (Rindell et al., 2019). This serves as the basis for this comparative study.

A variety of technologies may be employed to calculate security debt and prioritize vulnerabilities, each with its own set of features and capabilities.

Tool Categories

This section helps you categorize tools based on their functions, making it easier to understand the wide range of tools available and choose the most appropriate one for your specific needs.

Static Analysis Tools

Static analysis tools are software development tools that analyze source code without running it. They examine the code for errors, security vulnerabilities, and coding standards breaches. Static analysis contributes to better software quality and security by discovering these issues early in the development cycle. These technologies are critical for maintaining high software development standards and can assist in avoiding costly mistakes and vulnerabilities from entering production.

- **SonarQube**, it is an open-source platform for continuous inspection of code quality and security, including vulnerability detection (Marcilio et al., 2019).

- **Checkmarx**, it is a commercial static application security testing (SAST) tool for identifying vulnerabilities in code (Singh, 2024).

- **Fortify on Demand (Micro Focus),** it is a commercial SAST tool for finding security vulnerabilities, coding errors, and compliance issues (Hellström, 2009).

Dynamic Analysis Tools

Dynamic analysis tools analyze software's behaviour during runtime. They run the code in controlled conditions and watch how it interacts with the environment. This enables them to detect issues like memory leaks, performance, and security flaws that might not be seen during static analysis. Dynamic analysis effectively evaluates real-world scenarios and ensures software works as intended under varying settings.

- **Burp Suite**, it is a commercial web application security testing (WAST) platform for intercepting, inspecting, and modifying web traffic (Kim, 2020).

- **OWASP ZAP**, it is an open-source WAST tool for finding vulnerabilities in web applications (Jakobsson& Häggström, 2022).

- **Nessus (Tenable)**, it is a commercial vulnerability scanner for identifying vulnerabilities in systems and networks (Holcomb, 2009).

- **OpenVAS**, it is an open-source vulnerability scanner for detecting vulnerabilities in systems and applications (Rahalkar, 2019).

- **Qualys**, it is a commercial vulnerability management and compliance solution (Sharma et al., 2024).

Risk Assessment Tools

Risk assessment tools are software applications that assist organizations in identifying, analyzing, and evaluating risks. These tools frequently use a variety of approaches, like threat modelling, vulnerability scanning, and impact analysis, to determine the probability and impact of certain risks. These tools assist organizations in making decisions regarding security measures and emergency plans. Risk assessment techniques are essential for protecting valuable assets and managing potential risks.

- **RiskLens**, it is a commercial platform for quantitative risk assessment and management (Barlow et al., 2021).

- **Security Scorecard**, it is an open-source tool for assessing and improving security (Arntzen Toftegaard, 2022).

- **ThreatModeler (Microsoft),** it is a commercial software that detects and mitigates security vulnerabilities (Pai & Kunte, 2022).

Security Debt Management Platforms

Security debt management systems are software tools that assist organizations in tracking security vulnerabilities. These platforms frequently interface with various security tools, including vulnerability scanners and static analysis tools, to offer a complete picture of an organization's security posture. By providing a disciplined approach to addressing security vulnerabilities, these systems help organizations reduce risk exposure and improve overall security posture.

- **DefectDojo**, it is an open-source vulnerability management application for tracking, managing, and prioritizing vulnerabilities (Bernardo, 2022).

- **Snyk**, it is a security tool that finds and resolves vulnerabilities in code, containers, and open source dependencies (D et al, 2023).

- **GitLab,** the GitLab DevOps platform has built-in security measures like vulnerability detection and licensing compliance (Rahman,2024).

Specialized Tools

Specialized tools are created for specialized jobs or industries and have more advanced features and capabilities than general-purpose tools. To utilize these technologies properly, you will likely need specialized expertise or training. Examples of specialized tools are:

- **Veracode Software Composition Analysis (SCA),** it is a commercial tool that identifies open-source vulnerabilities and license compliance concerns (Singh, 2024).

- **Black Duck (Synopsys),** it is a commercial SCA tool for managing open-source risk (Lallet et al., 2008).

- **OWASP Dependency-Check**, it is an open-source tool for identifying and assessing vulnerabilities in open-source components (Cadariu et al., 2015).

- **FOSSA**, it is a commercial open-source software management platform with vulnerability scanning capabilities (Zhang, 2020)

- **Aqua Security**, it is a commercial platform for securing containerized applications, including vulnerability scanning and compliance (Makani& Jangampeta, 2024).

- **Trivy**, it is an open-source vulnerability scanner for container images (Zarei, 2022).
- **SQLError**, it is an open-source tool for detecting SQL vulnerabilities in applications (Cebollero et al., 2015).

This study will evaluate these tools to discover the best solutions for diverse project contexts and organizational demands.

Selection Criteria: For Security debt management

This research focuses on reducing security debt. To achieve this goal, the selection of tools was prioritized based on their relevance to effective security management, as outlined below:

- **Security Debt Metrics Analysis**, they are tools were evaluated based on their capability to analyze critical security debt metrics. This contains functionality for evaluating vulnerability fix time, cost, and risk reduction. These measurements are critical in determining the exact impact of security debt and the impact of reduction. These metrics included:
  - **Vulnerability Fix Time**, it is the average time required to repair a vulnerability, from discovery to resolution.
  - **Cost** is the financial resources needed to repair a vulnerability, including labour, equipment, and potential business effects.
  - **Risk reduction**, it is the decrease in the occurrence and impact of a security event once a vulnerability has been fixed.

These criteria were chosen as they give a complete picture of the impact of security debt and the effect of repairing activities. The vulnerability fix time aids in the identification of slow-moving vulnerabilities, but the cost of repair influences resource allocation decisions. Risk reduction assesses the total effectiveness of security debt mitigation.

- **Usability for Security Debt Management**, while usability is essential, the selection process takes into consideration the special requirements of security debt management. Priority is given to tools that are easy to use and understand, especially for stakeholders with less technical expertise. Throughout the study process, this ensures informed decision-making and active participation.

By keeping these standards in mind, the tools chosen will be ideal for evaluating and contrasting different security debt reduction approaches, eventually yielding insightful findings for this study.

Selected SCA Tools Based on Selection Criteria

After defining the selection criteria and objectives, the following potential tools were selected for comparison, as they show their applicability in security reduction:

- **Black Duck (Synopsys)**, it automates open-source security and licensing compliance for developers and security teams. While its primary job is to manage open-source components, it indirectly tackles technological debt by identifying and addressing possible risks early in the development process. Key metrics addressed are likely to include vulnerability numbers, license compliance status, and maybe code coverage for security checks (Lallet et al.,2008).

- **Sonatype Nexus**, Nexus is primarily a software component repository manager that includes security tools for identifying component vulnerabilities. It helps to reduce technological debt by organizing component management and implementing security checks. Metrics might include vulnerability numbers, dependency management efficiency, and even licensing compliance (Vojnović, 2023).

- **Veracode SCA**, it is specifically developed to examine code for licensing conflicts and open-source vulnerabilities, addressing technological debt by identifying possible security issues early in the development process. Typical metrics include vulnerability numbers, license compliance status, and possibly code coverage connected to vulnerability scans (Singh, 2024).

- **Snyk** is a cloud-native open-source security platform That provides vulnerability scanning and code repair. It contributes to reducing technical debt by giving rapid feedback on vulnerabilities and remedy choices. Metrics would most likely include vulnerability counts, repair rates, and maybe code quality metrics linked to security (D et al., 2023).

- **OWASP Dependency-Check**, this open-source program detects open-source components and scans for known vulnerabilities. While primarily concerned with vulnerability detection, it helps to reduce technological debt by flagging potential security issues. Vulnerability numbers and dependency management efficiency are two potential metrics (Cadariu et al., 2015).

- **FOSSA**, it is a cloud-based platform for managing open-source dependencies and detecting security issues. It contributes to the reduction of technological debt by making open-source components and their associated risks visible. Metrics might include vulnerability numbers, license compliance status, and a review of open-source component use (Zhang, 2020).

- **Aqua Security**, which specializes in containerized application security, indirectly reduces technological debt by securing essential elements. Metrics would most likely include vulnerability numbers in container images, compliance status, and maybe deployment speed (Makani& Jangampeta, 2024).

- **SQLError**, it is specifically developed to detect SQL injection vulnerabilities, addressing technological debt by flagging a common security issue. Metrics would primarily focus on SQL injection vulnerability numbers and, maybe, code coverage for SQL injection checks (Cebollero et al.,2015).

- **Fortify on Demand (Micro Focus)**, it is a cloud-based tool for application security testing that includes SCA. It helps decrease technical debt by discovering and fixing vulnerabilities. Metrics will likely include a wide range of security vulnerabilities, code quality metrics, and maybe compliance status (Hellström, 2009).

- **Trivy**, it is vulnerable package scanner for container images. It solves technological debt by detecting possible security concerns in containerized systems. Metrics would primarily focus on vulnerability numbers in container images and maybe image creation efficiency (Zarei, 2022).

Software Composition Analysis Tool Comparison

Software Composition Analysis (SCA) tools are crucial for identifying and managing software vulnerabilities. To assist organizations in choosing the best option, this research assesses several popular SCA tools, outlining their strengths and weaknesses and recommended usage.

Key Features and Capabilities of SCA Tools

SCA tools include several features to help you find and fix software vulnerabilities. Here is an overview of some important tools:

- **Black Duck**, Black Duck's SCA and SBOM (Software Bill of Materials) development capabilities assist organizations in efficiently managing open-source components. This lowers security debt by detecting vulnerabilities early in the development process and helps with licensing compliance by monitoring utilized components and licenses (Lallet et al., 2008).

- **Sonatype Nexus**, while Nexus is essentially repository management, its integrated SCA features assist in identifying vulnerabilities in components stored within the repository. This helps to reduce security debt by identifying risks before deployment. However, handling external dependencies might require extra tools (Vojnović, 2023).

- **Veracode**, it offers complete SCA in addition to other security testing technologies, providing a complete solution to application security. This helps reduce security debt by detecting various vulnerabilities and verifying compliance through licence checks (Singh, 2024).

- **Snyk**, snyk's extensive SCA capability and smooth integration with CI/CD pipelines enable early vulnerability identification. This considerably reduces security debt by moving it to the left. Automated vulnerability identification and fix ideas assist in more efficient debt reduction (D et al., 2023).

- **OWASP Dependency-Check**, a lightweight open-source tool, is successful in identifying vulnerabilities in open-source components. While its major aim is vulnerability identification, it also helps to reduce security debt by noticing possible threats (Cadariu et al.,2015).

- **FOSSA**, it focuses on open-source SCA and licensing compliance and assists organizations in managing open-source components in a secure ethical manner. By discovering vulnerabilities and licence breaches, FOSSA helps to reduce security debt and ensure compliance (Zhang, 2020).

- **Aqua Security**, Aqua Security's container security platform now includes SCA, providing full protection for cloud-native settings. This helps reduce security debt by addressing vulnerabilities in container images, a significant component in modern systems (Makani& Jangampeta, 2024).

- **SQLError**, it is specifically built to address SQL injection vulnerabilities, a common security problem. By detecting these vulnerabilities, we may reduce security debt and increase application security (Cebollero et al., 2015).

- **Fortify on Demand**, with its strong SCA capabilities and integration with development lifecycles, allows for early vulnerability identification and mitigation. This reduces security debt by detecting and resolving vulnerabilities throughout the development phase (Hellström, 2009).

- **Trivy:** a lightweight, open-source vulnerability scanner for container images, is successful at identifying possible vulnerabilities. This helps reduce security debt by assuring the security of containerized applications (Zarei, 2022).

Understanding the major characteristics and capabilities of different SCA tools allows organizations to pick the best solutions for their unique security and compliance requirements, eventually reducing security debt and enhancing overall application security.

Strengths and Weaknesses of SCA Tools

It's important to know the SCA tool's strengths and weaknesses with respect to the project's needs before selecting it.

Snyk, Veracode, and Fortify on Demand

- **Snyk**, it provides wide vulnerability detection, easy interaction with CI/CD pipelines, and prioritization features. While these advantages contribute to quick security debt reduction, complex systems may necessitate additional configuration work (D et al., 2023).
- **Veracode,** it offers complete SCA and security testing tools. This comprehensive strategy can effectively reduce security debt. However, the price approach may be too expensive for large-scale implementations (Singh, 2024).
- **Fortify on Demand**, it provides comprehensive code analysis and integration into development workflows. This can result in earlier vulnerability detection and reduced security debt. However, the subscription approach may not be appropriate for smaller companies with low funding (Hellström, 2009).

FOSSA and OWASP Dependency Check

- **FOSSA**, it helps managing open-source licenses, therefore reducing legal concerns and associated security consequences. However, vulnerability detection skills may be more limited than commercial options, thereby affecting security debt reduction (Zhang, 2020).
- **OWASP Dependency-Check**, it is a free and open-source solution for identifying fundamental vulnerabilities. While it can be a useful starting point for reducing security debt (Cadariu et al., 2015).

Black Duck and Sonatype Nexus

- **Black Duck**, it offers robust SBOM development and integration capabilities, which help in supply chain security and compliance. However, managing complicated projects can be difficult, affecting efficiency and security debt reduction (Lallet et al., 2008).
- **Sonatype Nexus**, it provides powerful repository and product management capabilities. While useful for component management, it may need the use of extra tools for external dependency management, affecting the overall security posture (Vojnović, 2023).

Aqua Security and Trivy

- **Aqua Security**, it offers full container security, such as vulnerability screening, compliance checks, and runtime protection. This can greatly reduce security debt in cloud-native situations. However, it may be overkill for standard applications (Makani& Jangampeta, 2024)
- **Trivy**, it provides lightweight container image scanning, making it ideal for fast vulnerability evaluation. While it is useful for basic assessments, it may fall short when compared to commercial products in terms of vulnerability complexity (Zarei ,2022).

SQLError

It is designed specifically for SQL injection vulnerabilities, and SQLError successfully fights this threat. However, its reach is restricted, and it may not be enough to meet larger SCA standards (Cebollero et al., 2015).

The following table presents a comparative analysis of various software composition analysis (SCA) tools. It highlights the key features and capabilities of each tool, enabling readers to select the most suitable SCA solution for their needs.

Table 2

Comparison Between SCA Tools

Tool	Vulnerability Scanning	License Management	CI/CD Integration	SBOM Generation	Open-source	Container Scanning	SAST Capabilities	Cloud-based
Black Duck (Synopsys)	Yes	Yes	Yes	Yes	No	No	No	Yes
Sonatype Nexus	Yes	Limited	Yes	No	No	No	No	Yes
Veracode SCA	Yes	Yes	Yes	No	No	Yes	Yes	Yes
Snyk	Yes	Basic	No	No	Yes	No	No	Yes
OWASP Dependency-Check	Yes	No	Yes	No	Yes	No	No	Yes
FOSSA	Yes (Limited)	Advanced	Yes	No	Yes	Yes	No	Yes
Aqua Security	Yes	No	Yes	No	Yes	Yes	No	Yes
SQLError	No	No	No	No	No	No	No	No
Fortify on Demand (Micro Focus)	Yes	No	Yes	No	No	Yes	No	No
Trivy	Yes	No	Yes	No	Yes	Yes	Yes	Yes

***Source*:** Prepared by the authors.

As shown in Table 2, a wide range of SCA tools are available to meet the requirements of various organizations. While Snyk, Veracode, and Fortify on Demand provide extensive functionality, their cost and complexity may make them unsuitable for smaller applications. FOSSA and OWASP Dependency- Check shine in open-source management, while commercial solutions frequently provide more complete vulnerability detection. Black Duck and Sonatype Nexus go beyond basic SCA; however, they may require more tools or incur greater expenses. Aqua Security and Trivy dominate container security, but SQLError addresses a specific database problem.

Finally, the best SCA tool depends on an organization's requirements, budget, and development environment.

Selection for the Most Appropriate SCA Tools

This section explores how different types of companies may use Software Composition Analysis (SCA) tools to meet their security requirements.

Applications of SCA Tools

This section explains how organizations in the real world employ software composition analysis (SCA) techniques to handle security concerns. As illustrated in Table 3, the application of SCA approaches can reduce security debt. The specific use cases listed in Table 3 illustrate how these technologies can be used to solve various security problems. Table 3 suggests software composition analysis (SCA) solutions for various domains depending on industry, project size, and security needs. It advises on picking the best SCA tool to meet certain security concerns.

Table 3

Choosing the Right SCA Tool: A Practical Guide

Tool	Suitable Fields	Justification	Reference
Black Duck (now part of Synopsys)	General SCA across industries (e.g., telecommunications, manufacturing, healthcare)	Black Duck's extensive feature set, which includes vulnerability detection, licensing compliance, and software composition analysis, makes it an adaptable alternative for a variety of organizations. The ability to manage large-scale projects and interact with numerous development tools is very useful in complicated contexts.	(Lallet et al.,2008)
Sonatype Nexus	Software development, DevOps (e.g., technology, finance, retail)	Sonatype Nexus excels in software component management and effectively adds security measures into the development cycle. Its focus on product dependency management makes it ideal for DevOps organizations and teams that value speed and efficiency.	(Vojnović, 2023)
Veracode SCA	Software development, application security (e.g., technology, education, government)	Veracode SCA provides a powerful platform for static and dynamic application security testing, making it appropriate for organizations that value thorough vulnerability assessment. The focus on application security is consistent with businesses subject to tight, satisfying regulatory requirements.	(Singh, 2024)
Snyk	Software development, DevOps, cloud security (e.g.,	Snyk's cloud-native platform and focus on developer-first security make it ideal for current development patterns. Its strengths	(D et al., 2023)

	telecommunications, media & entertainment, automotive)	in container and cloud security are in line with the requirements of organizations using cloud-based architectures.	
OWASP Dependency -Check	Open-source projects, security researchers (all industries)	Dependency-Check is an open-source tool that is free to use for projects of any size. Its major focus on discovering vulnerabilities in open-source components makes it useful for organizations concerned with supply chain security.	(Cadariu et al.,2015).
FOSSA	Open-source projects, license compliance (all industries)	FOSSA excels at managing open-source licenses and ensuring compliance, making it important for organizations with complicated software supply chains. The ability to detect possible legal concerns linked with open-source usage is critical for risk management.	(Zhang, H., 2020)
Aqua Security	Containerized applications, cloud security (e.g., cloud providers, telecommunications, financial services)	Aqua Security offers complete security for containerized settings, including vulnerability screening, runtime protection, and compliance. Its emphasis on cloud-native apps makes it ideal for organizations implementing containerization plans.	(Makani& Jangampeta, 2024)
SQLError	Static application security testing (SAST) (all industries)	SQLError specializes in identifying SQL injection flaws, which are a serious security issue. Its emphasis on a single danger area makes it an invaluable tool for organizations that prioritize database security.	(Cebollero et al., 2015)
Fortify on Demand (Micro Focus)	Broad SCA use cases (e.g., technology, healthcare, government)	Fortify provides a complete platform for application security testing, which includes SCA, SAST, and DAST. Its extensive feature set and scalability make it ideal for organizations with complex security needs.	(Hellström, 2009)
Trivy	Container security, vulnerability scanning (e.g., cloud providers, DevOps, containerized applications)	Trivy is a lightweight and efficient vulnerability scanner for container images. Its emphasis on container security is consistent with the demands of organizations using containerization.	(Zarei, 2022)

Source: prepared by the authors

Result Analysis of Different SCA Tool Deployment

Deploying SCA technologies offers several advantages to organizations trying to improve their security posture. Here's a closer look at some key results:

- **Improved Vulnerability Detection**, SCA tools do more than identifying vulnerabilities. They may review the codebase to determine the specific location of the vulnerability, evaluate its impact and the potential to determine how serious it is, and even make recommendations for possible fixes. Developers can effectively identify vulnerabilities and prioritize fixes with the help of this complete information.

- **Reduced Security Debt**, by proactively resolving vulnerabilities identified by SCA tools, organizations may decrease their security debt and the attack surface for potential attackers.
- **Streamlined Patch Management**, SCA tools can automate vulnerability identification and prioritization, allowing development and security teams to focus on patching the most critical vulnerabilities first.
- **Increased Developer Productivity**, SCA tools may be integrated into development workflows to provide developers with real-time alerts about vulnerabilities in their code. This enables them to patch vulnerabilities earlier in the development cycle, eliminating rework and saving time and effort.
- **False Positives and Ongoing Challenges**, it is critical to recognize that SCA tools can occasionally provide false positives, requiring manual verification by security professionals. Additionally, updating SCA tools with the most recent vulnerability databases is critical to their usefulness.

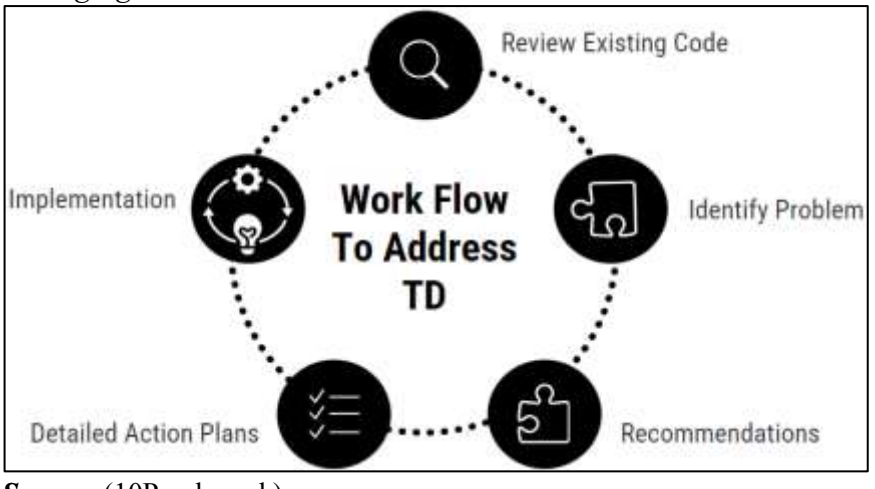
Conclusion and Future Directions

An organization's security posture is seriously threatened by security debt, which is the accumulation of unresolved security vulnerabilities. The negative consequences of security debt were discovered by this study, which include a higher chance of breaches, data loss, and reputational damage. Tools with features associated with security debt measures, data exchange, usability, and availability were given a lot of weight throughout the selection process. Following these criteria, a several practical choices were discovered, including commercial services like Snyk, Veracode, and Black Duck and open-source alternatives like OWASP Dependency-Check and Trivy. The ultimate decision will be based on how well these tools correspond with the specific study objectives and available resources. The study tries to give useful insights into the capabilities of various security debt reduction solutions. The findings will help organizations gain a better knowledge of how to pick and use these tools effectively to manage and reduce security debt.

Figure 6 shows the essential stages involved in effectively managing technical debt, including reviewing existing code, identifying problems, recommendations, detailed action plans and implementation of the solution. By following these steps, organizations can proactively address technical debt and improve the overall health and sustainability of their software systems. Addressing security debt may greatly enhance an organization's overall security posture. This results in better data protection, a lower chance of cyberattacks, and more compliance with security regulations.

Figure 6

Managing Technical Debt



Source: (10Pearls, n.d.).

Several aspects require more investigation to strengthen security debt management practices, such as:

- **Improved Measurement Techniques**, it is critical to develop more precise and consistent ways of measuring security debt. This would allow organizations to properly measure their security debt load and track how it is reduced over time.

- **Prioritization Frameworks**, which assist organizations in determining which security vulnerabilities to address first, would be extremely beneficial. These frameworks might take into consideration risk, possible damage, and simplicity of clean-up.

- **The Impact of New Technologies**, emerging technologies like automation and machine learning have the potential to revolutionize security debt management. More study is needed to determine how these tools may be used to automate vulnerability detection, patching methods, and security debt tracking (Dissanayake et al., 2022).

- **The Role of Security Culture**, creating a strong security culture inside organizations is critical for avoiding the build-up of security debt in the first place. Further study might look into ways to foster a culture of security knowledge, ownership, and continual improvement.

- **Recognizing the Consequences of Security**, organizations may enhance their overall security posture and minimize the risk of cyberattacks by recognizing the consequences of security debt and taking proactive actions to control it through improved assessment, prioritization, and the use of new technologies. Furthermore, cultivating a strong security culture may assist in avoiding the building of security debt in the first place, resulting in a more secure digital ecosystem.

References

10Pearls. (n.d.). Technical debt management https://10pearls.com/technical-debt-management/ Accessed 15/9/2024

Arntzen Toftegaard, Ø. A. (2022). An effect analysis of ISO/IEC 27001 certification on technical security of Norwegian grid operators. *In 2022 IEEE International Conference on Big Data (Big Data)*, 2620–2629. IEEE. https://doi.org/10.1109/BigData55660.2022.10020529

Barlow, C., Walklate, S., & Johnson, K. (2021). Risk refraction: Thoughts on the victim-survivor's risk journey through the criminal justice process. *International Journal for Crime, Justice and Social Democracy, 10*(3), 177-190. https://search.informit.org/doi/10.3316/informit.026564949450511

Bernardo, G. (2022). *DevSecOps pipelines improvement: new tools, false positive management, quality gates and rollback* (Master's Thesis, Politecnico di Torino). Politecnico di Torino.

Cadariu, M., Bouwers, E., Visser, J., & Van Deursen, A. (2015). Tracking known security vulnerabilities in proprietary software systems. *In 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 516-519. IEEE.

Cebollero, M., Natarajan, J., Coles, M. (2015). *Error handling and dynamic SQL*. In: Pro T-SQL Programmer's Guide. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-0145-9_18

Cifuentes, C., Gauthier, F., Hassanshahi, B., Krishnan, P., & McCall, D. (2023). The role of program analysis in security vulnerability detection: Then and now. *Computers & Security, 135*, 103463. https://doi.org/10.1016/j.cose.2023.103463.

Coetzer, C., and Leenen, L. (2024). Managing cyber security debt: strategies for identification, prioritization, and mitigation. *In Proceedings of 19th International Conference Cyber Warfare and Security (ICCWS), 19*(1), 439-446. doi: 10.34190/iccws.19.1.2178.

D, S., M K, N., Ashok Kumar, R., & Nidugala, M. (2023). To detect and mitigate the risk in continuous integration and continues deployments (CI/CD) pipelines in supply chain using Snyk tool. *In 2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, 1-10. DOI:10.1109/CSITSS60515.2023.10334136

Debricked. (2021, October 11). SCA Tools Overview. https://debricked.com/blog/sca-tools-overview/

Dissanayake, N., Jayatilaka, A., Zahedi, M. & Babar, *M.A. (2022).* An empirical study of automation in software security patch management. *In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 1-1.

Dudee, Y. (2021). *Exploration of technical debt in plan-based vs. agile processes: A standard literature review*. FinalPaper_TechnicalDebt-converted2 (1).pdf

Hellström, P. (2009). *Tools for static code analysis: A survey* [PhD Thesis, Linköping University-Department of Computer and Information Science]. Linköping University. https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-16658

Holcomb, J. (2009, May 11-12). Auditing cyber security configuration for control system applications. *In 2009 IEEE Conference on Technologies for Homeland* Security, 7-13. IEEE. https://doi.org/10.1109/THS.2009.5168008

Imtiaz, N., Thorn, S., & Williams, L. (2021). A comparative study of vulnerability reporting by software composition analysis tools. *In Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1-11. IEEE. https://doi.org/10.1145/3475716.3475769

Itech India. (2021, June 29). Is technical debt the monster it is made out to be? https://itechindia.co/us/blog/is-technical-debt-the-monster-it-is-made-out-to-be/

Izurieta, C., Rice, D., Kimball, K., & Valentien, T. (2018). A position study to investigate technical debt associated with security weaknesses. *In Proceedings of the 2018 International Conference on Technical Debt*, 138-142. https://doi.org/10.1145/3194164.3194167

Jakobsson, A., & Häggström, I. (2022). *Study of the techniques used by OWASP ZAP for analysis of vulnerabilities in web applications* (Master's Thesis, Linköping University - Department of Computer and Information Science). Linköping University. https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-186346

Kim, J. (2020). *Burp suite: Automating web vulnerability scanning* [Master's Thesis, Utica College]. Utica College. Burp Suite: Automating Web Vulnerability Scanning - ProQuest

Kruke, M. M. (2022). Security debt in practice: *A qualitative case study* (Master's thesis, University of Oslo). University of Oslo. masterthesis-maren-maritsdatter-kruke.pdf (uio.no)

Lallet, J., Pillement, S., & Sentieys, O. (2008). Efficient dynamic reconfiguration for multi-context embedded fpga. *Proceedings of the 21st annual symposium on Integrated circuits and system design*, 210-215. DOI:10.1145/1404371.1404428

Li, Y., Soliman, M. and Avgeriou, P. (2023). Automatic identification of Self-admitted technical debt from four different sources. *Empirical Software Engineering, 28*(3), 65. https://doi.org/10.1007/s10664-023-10297-9

Makani, S. T., & Jangampeta, S. (2024). Devops security tools evaluating effectiveness in detecting and fixing security holes. *International Journal of DevOps (IJDO), 1*(2), 1-12.

Marcilio, D., Bonifácio, R., Monteiro, E., Canedo, E., Luz, W., & Pinto, G. (2019). Are static analysis violations really fixed? A closer look at realistic usage of SonarQube. *In 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, 209–219. IEEE. https://doi.org/10.1109/ICPC.2019.00040

Martinez, J., Quintano, N., Ruiz, A, Santamaria, I., De Soria, I., & Arias, J. (2021). Security debt: Characteristics, product life-cycle integration and items. *In Proc. 2021 IEEE/ACM International Conference Technical Debt (TechDebt),* 9-18. IEEE. doi: 10.1109/TechDebt52882.2021.00009.

McClintock, M. (202). 8 common sources of security debt (& tips to address each). Better AppSec. https://betterappsec.com/8-common-sources-of-security-debt-tips-to-address-each-f42de8e45bb7

Melo, A., Fagundes, R., Lenarduzzi, V., & Santos, W.B. (2022). Identification and measurement of requirements technical debt in software development: A systematic literature review. *Journal of Systems and Software, 194*(c), 111483. https://doi.org/10.1016/j.jss.2022.111483

Olsson, J., Risfelt, E., Besker, T., & Martini, A., Torkar, R. (2021). Measuring affective states from technical debt: A psychoempirical software engineering experiment. *Empirical Software Engineering. 26*. 105. https://doi.org/10.1007/s10664-021-09998-w

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

Pai, S & Kunte, S. (2022). A Comprehensive analysis of automated threat modelling solution company: Threat modeler software, Inc. *In International Journal of Case Studies in Business, IT and Education (IJCSBE), 6*(2), 99–107. DOI:10.47992/IJCSBE.2581.6942.0186

Rahalkar, S. (2019). *OpenVAS.* In: Quick start guide to penetration testing. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4270-4_2

Rahman, M. M., Barek, M. A., Akter, M. S., Riad, A. K. I., Rahman, M. A., Shahriar, H., Rahman, A., & Wu, F. (2024). Authentic learning on DevOps security with labware: Git hooks to facilitate automated security static analysis. *In Proc. of 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2418-2423. IEEE.

Rindell, K., Bernsmed, K., & Jaatun, M.G. (2019). Managing security in software: Or: How I learned to stop worrying and manage the security technical debt. In Proceedings of the 14th International Conference on Availability, Reliability and Security, 1-8. DOI:10.1145/3339252.3340338

Rios, N., Neto, M.G. and Spínola, R.O. (2018). A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology, 102*, 117-145.

Sharma, M., Desai, D., Arun, A. R., L. P., & Rajagopalan, N. (2024). OpenVAS vs the rest: Unveiling the competitive edge in vulnerability scanners. *In 2024 3rd International Conference for Innovation in Technology (INOCON)*, 1-6. IEEE. https://doi.org/10.1109/INOCON60754.2024.10511864

Siavvas, M., Tsoukalas, D., Jankovic, M., Kehagias, D., Chatzigeorgiou, A., Tzovaras, D., Anicic, N., Gelenbe, E. (2019). *An empirical evaluation of the relationship between technical debt and software security*. 10.13140/RG.2.2.15488.79365

Singh, A. (2024). Microservices security vulnerability remediation approach using Veracode and Checkmarx. *Journal of Artificial Intelligence General Science (JAIGS), 4*(1), 145–151. https://doi.org/10.60087/jaigs.v4i1.128

Stepsize. (2021). *The state of technical debt 2021: What engineers think about technical debt and its impact on team morale, velocity, and customer experience (Survey*). https://assets.website-files.com/5f922f81cc30586744dc7122/60e306c6db6224328eaf47a3_Tech%20debt%20report.pdf

TatvaSoft Blog. (2023, December 12). Guide to software composition analysis. https://www.tatvasoft.com/outsourcing/2023/12/software-composition-analysis.html

Vojnović, J. (2023). Mitigating supply chain attacks through detection of high-risk software dependencies. J_Vojnovic___Mitigating_supply_chain_attacks_through_detection_of_high-risk_software_dependencies.pdf (ru.nl)

Zaimi, A., Ampatzoglou, A., Triantafyllidou, N., Chatzigeorgiou, A., Mavridis, A., Chaikalis, T., Deligiannis, I., Sfetsos, P., Stamelos, I. (2015). An empirical study on reusing third-party libraries in open-source software development. *In Proceedings of the 7th Balkan Conference on Informatics Conference (BCI '15)*. Association for Computing Machinery, 1–8. https://doi.org/10.1145/2801081.2801087

Zarei, M. (2022). Investigating the inner workings of container image vulnerability scanners (Master's thesis, Oslo Metropolitan University). https://oda.oslomet.no/oda-xmlui/bitstream/handle/11250/3017416/zarei-acit2022.pdf?sequence=1&isAllowed=y

Zhang, H. (2020). Comparison of open source license scanning tools. Bachelor Degree Project. FULLTEXT01.pdf (diva-portal.org)

المجلة الدولية للسياسات العامة في مصر - مجلد 3 - العدد (4) - أكتوبر 2024
ISSN: Print: 2812 - 4758 , Online: 2812 - 4766
تصدر عن مركز المعلومات ودعم اتخاذ القرار

التقييم المقارن لأدوات تحليل تكوين البرمجيات في سياق تخفيض الديون الفنية

المستخلص

يُستخدم مصطلح "الدين الفني" في هندسة البرمجيات لوصف الحلول التقنية التي قد تكون عملية على المدى القصير ولكن لها عواقب ضارة على المدى الطويل. تركز الدراسة على تقييم أدوات تحليل تكوين البرمجيات Software Composition Analysis (SCA) في سياق تقليل الديون الفنية، وخاصة ديون الأمان، فالديون الفنية هي عواقب القرارات السريعة التي تؤدي إلى مشاكل في الصيانة والأمان في المستقبل. وتسلط الدراسة الضوء على أهمية أدوات SCA في ممارسات تطوير البرمجيات الحديثة من خلال تحديد نقاط الضعف في المكونات مفتوحة المصدر ومعالجتها بشكل فعال، ويمكن لهذه الأدوات تقليل الديون الفنية بشكل كبير وتحسين الأمان والجودة العامة للتطبيقات البرمجية. وقارنت الدراسة بين أدوات SCA المختلفة بناءً على قدراتها وسهولة استخدامها وفعاليتها في تقليل الديون الأمنية. وتوصلت الدراسة إلى أن أدوات SCA فعالة في تقليل الديون الفنية، خاصة ديون الأمان، ويجب على المؤسسات عند اختيار أداة SCA مراعاة نوع التطبيق، وطرق التطوير، والموقف الأمني للمنظمة.

الكلمات الدالة: الديون الفنية، تحليل تكوين البرمجيات (SCA)، البرمجيات مفتوحة المصدر (OSS)، حل التبعية، هندسة النظم