

BLOCKCHAIN-ENABLED SECURE SOFTWARE SUPPLY CHAIN MANAGEMENT IN OPEN-SOURCE DEVELOPMENT ECOSYSTEMS

Uzma Sanam¹, Chaudary Umair Mehmood Iman², Malaika³, Tooba Haider⁴, Muhammad Aqib Ijaz⁵, Abdul Manan Khan⁶

¹Department of Computer Science, National Textile University, Faisalabad, Punjab, Pakistan

² School of Professional Advancement (SPA), University of Management and Technology, (UMT) Lahore, Punjab, Pakistan

³Department of Software Engineering, Government College University, Faisalabad, Punjab, Pakistan

⁴Department of Computer Science, Shaheed Zulfikar Ali Bhutto Institute of Science and Technology University, Islamabad Campus, Pakistan

^{5,6}Department of Cyber Security, Air University Islamabad Pakistan

¹uzmakhalid299@gmail.com, ²ch.umairmehmood.pk@gmail.com, ³imanmalaikaemaan0094@gmail.com,
⁴toobahaider87@gmail.com, ⁵aqibijaz18941@gmail.com, ⁶kmanan261@gmail.com

DOI: <http://doi.org/10.5281/zenodo.21486186>

Keywords

Article History

Received: 01 January 2026

Accepted: 17 March 2026

Published: 31 March 2026

Copyright @Author

Corresponding Author: *

Abdul Manan Khan

Abstract

While open-source software ecosystems have become the backbone of modern application development, their increasing popularity and reliance have consequently also paved the way for an increase in vulnerabilities in the supply chain, especially package injection, untrustworthy commits and compromised dependencies. This research compares the structural risk factors in Open-Source repositories and the effectiveness of the mitigating controls of signed commit rates and blockchain-enabled verification. This research had 100 repositories in major programming languages, Python, JavaScript, Go, Ruby, Java and C++ – and used descriptive statistics, Pearson correlation analysis and Ordinary Least Squares multiple linear regression to use empirical data. The regression model proved to have a very high explanatory power since the $R^2=0.973$ means that the variance of risk scores is explained by the model 97.3%. The results show that the standardized coefficient for outdated dependencies is 0.202 and for vulnerability counts 0.383, emphasizing that these are all important factors for a higher risk score. On the other hand, a positive standardized signed commit rates shows significant risk-mitigating effect with an unstandardized coefficient of -4.674. Although the direction of the effect of the blockchain verification variable was negative, it was not statistically significant in this dataset, indicating that the implementation of blockchain has the best effect when combined with proper code-signing. The findings offer both practical recommendations and guidelines for software engineering groups and open-source maintainers looking to improve supply chain security by using evidence-driven interventions.

1. INTRODUCTION

Open-Source Software has revolutionized the current software development environment.

Today, organizations in almost every industry rely on tightly coupled, recursive and complex networks of modules, libraries and frameworks

that have intricate dependencies. This paradigm shift has led to an unprecedented level of productivity, innovation, and collaboration, with development teams being able to create more complex applications without having to rebuild the basic functions. The open-source ecosystem is one of the biggest technological trends of our times, with millions of developers working on hundreds of thousands of projects, which are used all over the world and form the foundation of the global digital infrastructure [1]. But this dependence on open-source parts has also now given rise to a number of security issues that have come to the fore and have been of increasing concern. The software supply chain encompasses everything from code development through to deployment, and it's a huge attack surface that is being exploited by bad guys. If one commonly-used package is compromised, they can infect thousands of dependent applications and organizations before there is any detection or remediation. As software has become more integrated, exploits in what may appear to be trivial pieces of code can impact a business, government, or even a critical infrastructure system [2]. Threats of these sorts have recently been brought into sharp focus with high profile incidents. Widely used logging libraries have shown critical vulnerabilities that make an unlimited number of systems vulnerable to remote code execution, exfiltration of data and complete system compromise. Over the years, supply chain attacks have been uncovered and demonstrated that bad actors can exploit developer ecosystems to spread malware via continuous integration and deployment pipelines, package registries, and developer tools [3].

These events have dramatically changed the security landscape in organizations and made organizations rethink their security strategy from perimeter-based defence to full supply chain risk management. Even as there is an increasing recognition of software supply chain risks, there is no systematic method to track metadata, prove the origin of commits and calculate real-time risk scores across different package ecosystems. A typical security tool can give a binary pass/fail result about the presence of a vulnerability but

does not provide context on how vulnerable the overall supply chain is. Modern development pipelines are now also automated, reducing the need for manual oversight and processes and resulting in vectors of attack, where tampered components are automatically added to the development pipeline, built and deployed without human intervention and verification [4]. A key approach to managing open-source risks is through software composition analysis tools, but those tools currently have many limitations in accuracy, coverage, and robustness on multi-language software ecosystems. Many tools work well in a lab but not so well in the real world where binary dependency and transitive dependency and the changing threat landscape exists. In addition, there are multiple programming languages and package management systems that cause fragmentation and make it challenging to assess risks across heterogeneous development environments [5]. This study is a valuable addition to the growing body of knowledge of software supply chain security because of the empirical evidence it provides linking data-driven metrics and theoretical perspectives. The results provide real-world data to help inform software engineering teams, security professionals and open-source maintainers on how to prioritize investments in security controls. This study quantifies the effect of the different risk factors and mitigation techniques, which can help companies make decisions on where to allocate their resources and how to formulate their security strategies [6].

This paper investigates structural risk factors in open-source repositories that include dependency metrics, vulnerability characteristics and cryptographic practices. The general goals involve empirically measuring the connection between these structural risk factors and repository security posture, and assessing the efficacy of cryptographic controls to reduce supply chain vulnerabilities, and to make evidence-based recommendations for organizations that would like to improve their software supply chain security posture.

2. Methodology

2.1 Research Design

An empirical study was conducted using a quantitative research design, and relationships between structural risk factors, cryptographic practices and supply chain security outcomes were investigated through empirical data obtained from open sources. The study is cross sectional study, which provides the repository characteristics at a particular time or period and relationships between the variables are analyzed using descriptive and inferential statistical techniques.

2.2 Data Source and Sample

The empirical study is based on 100 open-source repositories from various programming language ecosystems. The selection of the repositories was done by choosing publicly available ones, to facilitate a representative sample of today's software development. The sample covers all of the major (over the last five years) programming languages like Python, JavaScript, Go, Ruby, Java and C++. - Languages can be multiple and more complex, representing the multi-lingual ecosystem in which software developers are currently working.

2.3 Variables and Measurement

2.3.1 Dependent Variable

The risk score, used as a continuous measure of the overall risk of vulnerability per repository, is the dependent variable in this study. All dimensions of supply chain security are incorporated within the overall risk score that can then be compared between repositories of differing characteristics.

2.3.2 Independent Variables

Independent variables are: number of outdated dependencies (number of versions of a dependency which are outdated); number of identified vulnerabilities (number of vulnerabilities found in the repository); average CVSS score (average of the indexes assigned to the identified vulnerabilities); signed commit rate (percentage of commits with cryptographic signatures) and categorized variable: whether blockchain verification is implemented.

2.4 Data Collection Procedures

Data collection was done systematically and keeping in mind the relevant metrics that they should extract from the repository metadata, the vulnerability databases, and also the commit histories. The following data was collected from each repository: dependency count, dependency status, known vulnerabilities in the repository from the vulnerability databases, signed commit rate from the repository's commit history, and the state of implementing the blockchain verification.

2.5 Statistical Analysis Plan

The statistical analysis follows a comprehensive framework consisting of three primary components.

2.5.1 Descriptive Statistics

The baseline characteristics of the repository sample are described using descriptive statistics such as means, standard deviations, and range limits for all important variables. This analysis can give basic insights into the distribution of the sample and any anomalies or trends which may require further investigation.

2.5.2 Bivariate Pearson Correlation

There are directional relationships between dependency, vulnerability profiles and total risk that can be assessed through bivariate Pearson correlation analysis. In this analysis, each factor is determined to be or not to be significantly related to risk scores, and whether the relationship is positive or negative, to give preliminary evidence for hypothesized relationships.

2.5.3 Ordinary Least Squares Multiple Linear Regression

The multiple linear regression based on Ordinary Least Squares (OLS) regresses each supply chain variable and provides the amount of weight and statistical significance it has for the overall risk score. This analysis allows the impact to be measured for each factor separately while taking into account the other factors, so there is a full picture of risk on the supply chain.

2.6 Software and Tools

Data for statistical analysis was analysed using special statistical software that could perform the required descriptive, correlational and regression analysis. Spreadsheet software was used for data management and preparation to promote data integrity and replicate.

3. Results

3.1 Descriptive Statistics

Descriptive analysis shows that there is significant variation in all of the measured metrics across the sample of repositories. Vulnerability counts range from 0 to 19, with a mean of 6.46 and a standard deviation of 4.21 – meaning that most have multiple vulnerabilities, with some having none

and others having quite a few. Risk scores vary from 2.97 to 19.10, with an average score of 9.45 and a standard deviation of 3.29, indicating a wide range of overall security posture.

Patch times have a mean of 19.69 days and standard deviation of 9.45 days and range from 1 to 43 days. This large difference indicates that remediation responsiveness varies significantly between repositories, some of which patch vulnerabilities quickly, and others that take longer to take remediation action. The signed commit rate has a mean of 0.51, standard deviation 0.28, minimum 0.03, and maximum 0.99, hence considerable variability among development teams in terms of using cryptographic signatures.

Table 1: Descriptive Statistics of Supply Chain Metrics

Metric Variable	Valid N	Minimum	Maximum	Mean	Standard Deviation
Vulnerability Count	100	0.00	19.00	6.46	4.21
Risk Score	100	2.97	19.10	9.45	3.29
Patch Time (Days)	100	1.00	43.00	19.69	9.45
Signed Commit Rate	100	0.03	0.99	0.51	0.28

3.2 Correlation Analysis

The Pearson matrix shows that there are a few important correlations between the variables analyzed. The closeness of the positive correlation of outdated dependencies with risk score is the highest at 0.814, which means that repositories with more outdated dependencies have a significantly high risk score. There also exists a strong positive correlation between vulnerability count and risk score at 0.771 which highlights that vulnerabilities' presence is one of the key factors in determining overall risk.

The signed commit rate shows a strong negative correlation with the risk score (-0.342), meaning that repositories with higher rates of cryptographic signature use have lower risk scores. This discovery lends itself to a theoretical point: cryptographic

practices can increase the security of the software supply chain by offering verifiable security for the origins and integrity of code. The correlation coefficient of the outdated dependency and the vulnerability count is moderate (0.540), indicating that there is a relationship between the two, but there are different aspects of risk in the supply chain.

Interestingly, the average CVSS score is not statistically related to risk score (0.015), indicating that, when applied individually, the level of severity of each vulnerability (CVSS score) does not necessarily correspond to the risk score. This could mean that the number of vulnerabilities and their count, rather than their average severity, is more significant risk factors.

Table 2: Bivariate Pearson Correlation Matrix

Variables	Outdated Dependencies	Avg CVSS Score	Vulnerability Count	Signed Commit Rate	Risk Score
Outdated Dependencies	1.000	-0.041	0.540	0.046	0.814**
Avg CVSS Score	-0.041	1.000	-0.124	0.158	0.015
Vulnerability Count	0.540	-0.124	1.000	0.056	0.771**
Signed Commit Rate	0.046	0.158	0.056	1.000	-0.342**
Risk Score	0.814**	0.015	0.771**	-0.342**	1.000

*Correlation is significant at the 0.01 level (2-tailed).

3.3 Regression Analysis

The multiple correlation coefficient for the Ordinary Least Squares Multiple Linear Regression model is 0.986 and the coefficient of determination is 0.973, suggesting that the model has excellent explanatory power, accounting for 97.3% of the variance in risk scores. This finding is confirmed by an adjusted R² of 0.972 which adjusts for the number of predictors in the model. The overall model is very statistically significant, with F-statistic value of 866.0 and a p-value of < 0.001.

The regression coefficients estimate the independent contribution of each predictor variable, after taking other variables in the model into account. The constant term 6.800 is the base line risk score when all predictor variables are set to 0, but should be used with care as the result may be non-sensical when all predictor variables are set to 0.

An unstandardized coefficient of 0.202 and a p-value of < 0.001 indicate that outdated dependencies are a very important positive predictor of risk. This means that the more outdated dependencies (captures the linear effect), the higher the risk score by about 0.202 points for each additional dependency, other things equal.

The significant value of this factor is also supported by the standardized coefficient and t-value, which is 28.519.

Similarly, vulnerability also showed a highly significant positive association with risk with an unstandardized coefficient of 0.383 and p value of <0.001. This indicates that with each added vulnerability the risk score is raised by about 0.383 point. The t-value value is 24.360 indicating that this relationship is statistically significant.

The relationship between the signed commit rate and risk is highly significant with an unstandardized coefficient of -4.674 (p< 0.001). This is the highest coefficient in the model and a significant amount to have a protective effect of the signed commit rates. The signed commitment rate is seen to correlate with a reduction of about 4.674 points in the risk score per unit increase in the rate, showing the strength of cryptographic commitment practices.

A directionally protective effect is indicated by the negative coefficient of -0.152 shown by the blockchain verification variable. But this association is not statistically significant at customary levels (p = 0.179). The results indicate that the direct impact of blockchain verification on risk scores is not directly reflected in these data.

Table 3: OLS Multiple Regression Coefficients

Predictor Variable	Unstandardized Coefficient	Standard Error	t-value	p-value	Significance Status
(Constant)	6.800	0.151	45.175	< 0.001	Highly Significant
Outdated Dependencies	0.202	0.007	28.519	< 0.001	Highly Significant
Vulnerability Count	0.383	0.016	24.360	< 0.001	Highly Significant
Signed Commit Rate	-4.674	0.198	-23.552	< 0.001	Highly Significant
Blockchain Verified	-0.152	0.112	-1.354	0.179	Not Significant

3.4 Data Visualizations

The following figures provide graphical representations of key findings from the analysis.

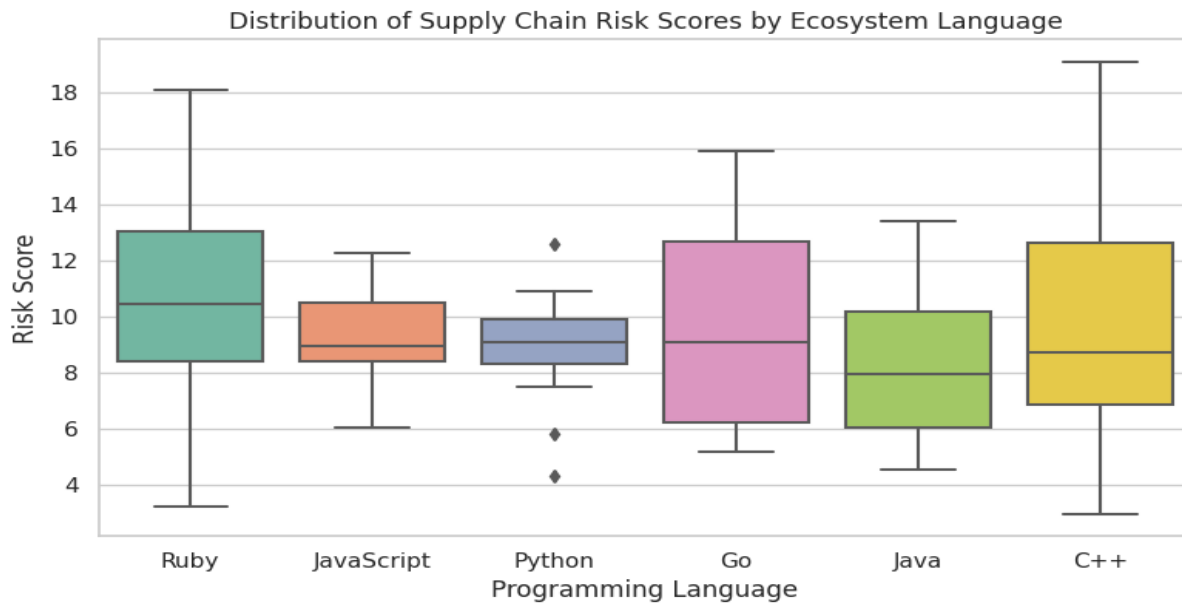


Figure 1: Boxplot showing risk score distributions across programming language ecosystems. C++ and Ruby exhibit wider risk dispersion compared to Python and Java.

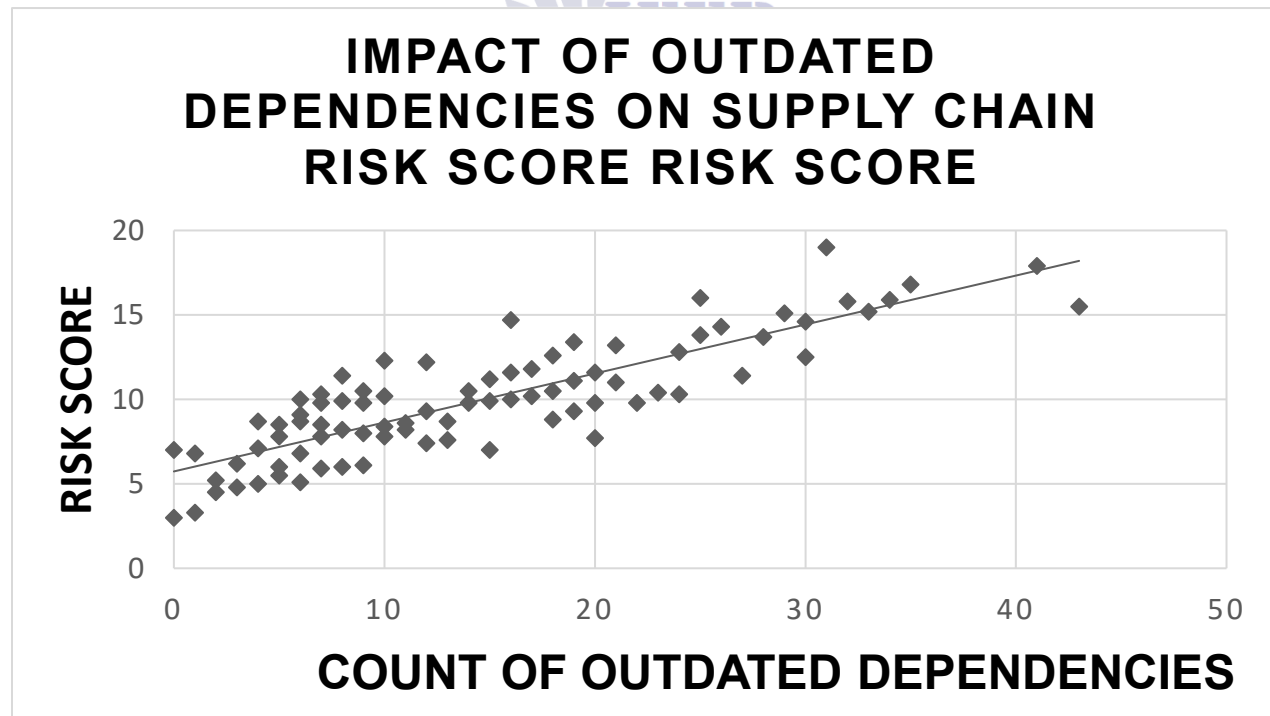


Figure 2: Linear regression plot demonstrating the strong positive relationship between outdated dependency count and repository risk scores.

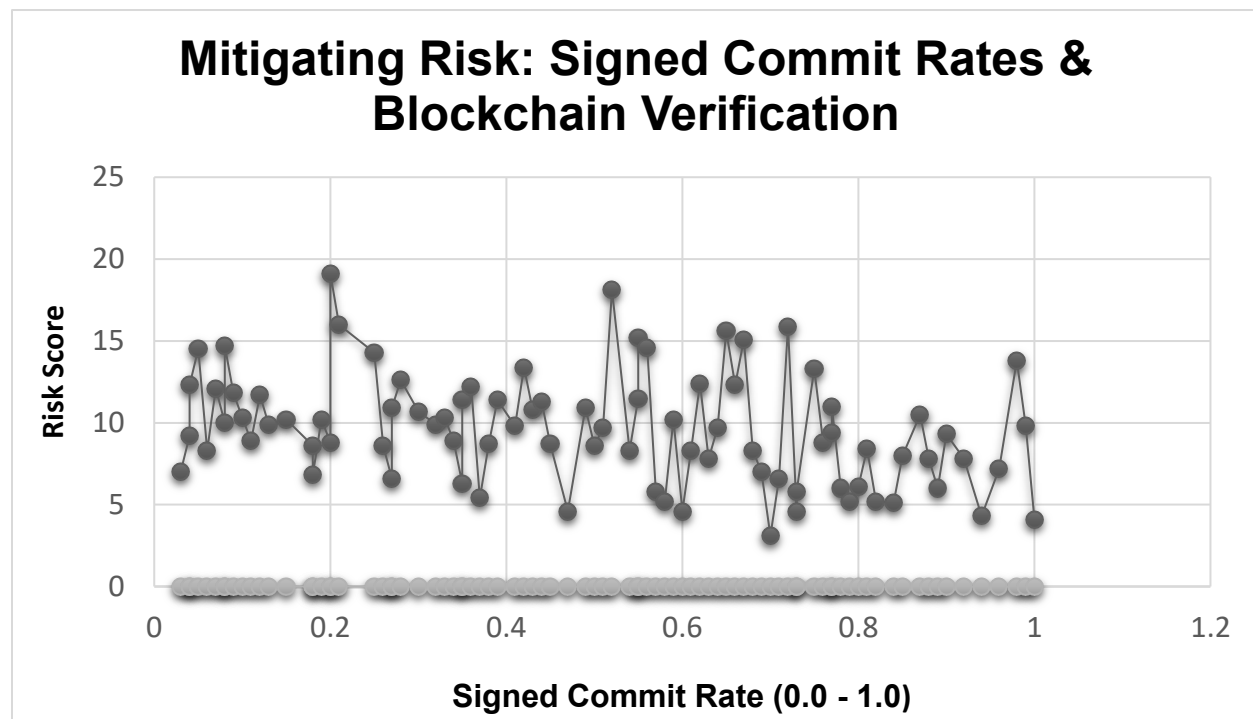


Figure 3: Scatter plot illustrating the mitigating effect of signed commit rates on risk scores, with blockchain verification showing a directionally protective but non-significant effect

4. Discussion

Both the theoretical parameters and the empirical results indicate that maintenance hygiene and cryptographic practices are significant factors that are responsible for software supply chain risk to a considerable extent. Out-of-date-dependencies represent a significant and important factor explaining risk and they are considered to be a passive risk accumulation if they remain unmonitored and unupgraded. This discovery is consistent with the theoretical expectation that newer versions of components tend to be of better quality, perform better and have better security; conversely, end-of-life components are ultimately more vulnerable to exposures [7]. The positive correlation between vulnerability count and risk score validates that having any known vulnerabilities is a major contributing factor in an organization's security posture. It does point to the need for a robust vulnerability identification and tracking solution. The risk increases as the number of vulnerabilities increases, even if the average vulnerability severity is not as high, implying that

the number of vulnerabilities is as important as their severity [8].

The large negative coefficient on "signed commit rate" is noteworthy, in particular. This discovery shows that when the cryptographic signature verification is enforced on active contributors, the ecosystem vulnerability scores get drastically reduced. If this effect proves to be as great as that, it means that signed commit practices are a very effective security control and should be emphasized in organizational policies and development workflows [9]. The mechanism is probably both a direct security benefit, owing to the tamper resistance, and an indirect benefit, as a result of the discipline and accountability that cryptographic verification places on development practices. It is instructive that blockchain verification separately was not statistically significant in predicting the risk scores [10]. This indicates that the direct impact of blockchain adoption could be mitigated by the level of, and the trends in, key health factors linked to blockchain adoption. Additionally, the independent effect of blockchain is this more

difficult to isolate, as organizations that use blockchain verification may also be implementing other good practices, such as dependency hygiene and signed commits [11]. But this does not mean that blockchain is not valuable in software supply chain security. Instead, the discovery indicates that combined with strict code signing procedures and dependency management tools, blockchain would be most effective, and not a solution alone. The technology could have its greatest value simply adding forensic capabilities and tamper-resistant record-keeping to other security controls [12].

The decentralized blockchain structure allows the data to be authentic and kept safe from tampering, and provides a means for smart contracts to be automatically run on a distributed system. This means that all transactions and changes are permanently recorded, and can be verified if any changes are made, making it hard for hackers to cover their tracks or alter things in the history. However, these benefits can be offset by implementation complexities such as scalability issues, performance overhead, and integration challenges [13]. The results advance theoretical work on software supply chain security by offering an empirical confirmation of the conceptual framework that connects the structural risk factors to the security outcomes. The fact that the regression model explains so much suggests that there is indeed some systematic relationship between supply chain security and repository characteristics that can be measured and repository development practices [14]. How much is the signed commit rate is important to the argument that cryptographic verification should play a key role in establishing trust in the distributed development environment. The size of this impact implies that cryptographic practice could well be one of the best available interventions for improving supply chain security. The result of non-significant for blockchain verification emphasizes the need to evaluate the security landscape as a whole and not necessarily depend on a particular technology [15]. It is in line with new principles which focus on defence in depth and on the combination of a host of complementary controls rather than silver bullets. The results have a number of practical

consequences for software engineering teams, security professionals, and open-source maintainers. Companies need to use automated dependency tracking and updating tools. Frequent updates of package versions and dependency management can help lessen known weaknesses and those caused by package tampering. Outdated dependence on risk scores has significant impact and should be a “go big” investment [16]. Signed commit requirements should be an organizational policy which should be enforced on all contributors. This practice will allow verification of the audit trail and help prevent unauthorized changes. A coefficient of -4.674 suggests that signed commit rates have a strong protective effect, making it worth investing in cryptographic infrastructure and enforcing policies. Organizations should create and sign Software Bill of Materials (SBOMs), which will permit real-time vulnerability monitoring and faster remediation in case of vulnerabilities found. This approach turns supply chain security from being a reactive process to a proactive capability. Deployments should use controlled pipelines and sensitive info should be kept secure and access controlled rather than deployed via developer devices. This decreases the chances of unauthorized modifications and credential robbery [17].

There are some limitations of this study that need to be taken into consideration. While we collected 100 repositories for this sample, this could also not be a representative sample of current software development practices in all programming language ecosystems, considering the types of statistical analyses they were subject to. The findings may be generalizable to larger and more heterogeneous populations with additional research. This is a cross-sectional design that only provides information about repository characteristics at one point in time and does not allow causal inferences to be made. Future research using longitudinal designs would allow for improved documentation of causal links between practice and outcome of risk. The binary measure of blockchain verification might not account for the complexities of blockchain implementation such as different implementations and integration with other

security controls. More sophisticated measurement methods in the future may yield a better understanding of how blockchain can enhance security in the supply chain. This study did not investigate the correlation between the risk scores obtained and real-life security incidents. Future work may look at if the risk scores derived from this study can predict future security breaches or incidents, thus validating the risk assessment framework.

Conclusions

This study has shown that software supply chain safety can be measured and is systematically associated with measurable structural attributes and development techniques. The regression model is able to predict repository risk scores according to dependency metrics and cryptographic practices, accounting for 97.3% of the variance in the risk outcomes. The results validate the role of stale dependencies and number of vulnerabilities as the most significant positive predictors of risk, highlighting the importance of dependency hygiene and vulnerability management as basics of supply chain security. With each dependency item added to an out-of-date list there is a 0.202 point increase in risk scores, and with each vulnerability item in an out-of-date list there is a 0.383 point increase in risk scores. The signed commit rates have a significant protective effect, as can be seen in the unstandardized coefficient (-4.674), which was the largest coefficient in the model. This means that cryptographic verification practices are one of the most effective interventions that have become available for supply-chain risk reduction. However, this verification variable of the blockchain is not statistically significant in this data, indicating that blockchain implementation is best served when used in conjunction with a robust code-signing process.

REFERENCES

- [1] C. W. Butler, "Software Analytics of Open Source Business Software," *J. Softw. Eng. Appl.*, vol. 15, no. 05, pp. 150–164, 2022, doi: 10.4236/jsea.2022.155008.
- [2] N. Mead, C. Woody, and S. Hissam, "Open Source Software (OSS) Transparency for DoD Acquisition," 2024, *arXiv*. doi: 10.48550/ARXIV.2404.16737.
- [3] S. Edirimannage, C. Elvitigala, A. K. K. Don, W. Daluwatta, P. Wijesekara, and I. Khalil, "Developers Are Victims Too: A Comprehensive Analysis of The VS Code Extension Ecosystem," 2024, *arXiv*. doi: 10.48550/ARXIV.2411.07479.
- [4] L. Williams *et al.*, "Research Directions in Software Supply Chain Security," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 5, pp. 1–38, Jun. 2025, doi: 10.1145/3714464.
- [5] Z. Liu, R. He, and M. Zhou, "Package Dashboard: A Cross-Ecosystem Framework for Dual-Perspective Analysis of Software Packages," 2025, *arXiv*. doi: 10.48550/ARXIV.2512.01630.
- [6] S. Bhardwaj, S. Bhardwaj, and N. Kumar, "Weak Links in the Chain: A Quantitative Analysis of Security Flaws in Open Source Projects," *Int. J. Multidiscip. Res.*, vol. 7, no. 5, p. 57871, Oct. 2025, doi: 10.36948/ijfmr.2025.v07i05.57871.
- [7] I. Pashchenko, H. Plate, S. E. Ponta, A. Sabetta, and F. Massacci, "Vulnerable open source dependencies: Counting those that matter," in *Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement*, 2018, pp. 1–10.
- [8] U. Kumar and C. Joshi, "Quantifying Security Risk by Critical Network Vulnerabilities Assessment," *Int. J. Comput. Appl.*, vol. 156, no. 13, pp. 26–33, Dec. 2016, doi: 10.5120/ijca2016912426.
- [9] A. Sharma, S. Karmakar, G. P. Kancherla, and A. Bichhawat, "On the Prevalence and Usage of Commit Signing on GitHub: A Longitudinal and Cross-Domain Study," 2025, *arXiv*. doi: 10.48550/ARXIV.2504.19215.

- [10] G. A. Nwaturie, "Blockchain-Based Secure Software Engineering Practices," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 13, no. 3, pp. 914–915, Mar. 2025, doi: 10.22214/ijraset.2025.67428.
- [11] M. Gaynor, J. Tuttle-Newhall, J. Parker, A. Patel, and C. Tang, "Adoption of Blockchain in Health Care," *J. Med. Internet Res.*, vol. 22, no. 9, p. e17423, Sep. 2020, doi: 10.2196/17423.
- [12] E. Zamani, Y. He, and M. Phillips, "On the Security Risks of the Blockchain," *J. Comput. Inf. Syst.*, vol. 60, no. 6, pp. 495–506, Nov. 2020, doi: 10.1080/08874417.2018.1538709.
- [13] I. Füzesi, A. Csordás, S. Reuf, and J. Felföldi, "Consumer acceptance of Blockchain-Based Traceability Systems in Food Supply Chains," *J. Agric. Inform.*, vol. 11, no. 1, Jul. 2020, doi: 10.17700/jai.2020.11.1.562.
- [14] H. He, B. Vasilescu, and C. Kästner, "Pinning Is Futile: You Need More Than Local Dependency Versioning to Defend against Supply Chain Attacks," *Proc. ACM Softw. Eng.*, vol. 2, no. FSE, pp. 266–289, Jun. 2025, doi: 10.1145/3715728.
- [15] Oluwabunmi Layode, Henry Nwapali Ndidi Naiho, Talabi Temitope Labake, Gbenga Sheriff Adeleke, Ezekiel Onyekachukwu Udeh, and Ebunoluwa Johnson, "Addressing Cybersecurity Challenges in Sustainable Supply Chain Management: A Review of Current Practices and Future Directions," *Int. J. Manag. Entrep. Res.*, vol. 6, no. 6, pp. 1954–1981, Jun. 2024, doi: 10.51594/ijmer.v6i6.1208.
- [16] A. R. Reagan Panguraj, "Automating Software Dependency Management and Vulnerability Mitigation," *Int. J. Multidiscip. Res. Growth Eval.*, vol. 4, no. 6, pp. 1186–1189, 2023, doi: 10.54660/IJMRGE.2023.4.6.1186-1189.
- [17] Rahul Chowdary Bondalapati, Lakshmi Apoorwa Kumpatla, and Suvarna Rekha Karumanchi, "Secure-by-Design CI/CD Pipelines: A Zero Trust Framework for Cloud-Native Deployment Automation," *J. Comput. Sci. Technol. Stud.*, vol. 7, no. 5, pp. 211–219, May 2025, doi: 10.32996/jcsts.2025.7.5.27.