Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 04, April-2025

Software Supply Chain Security: Designing a Secure Solution with SBOM for Modern Software EcoSystems

Osha Shukla

***Abstract*— Nowadays attacks are becoming more and more sophisticated. Supply chain attack is one of them. This attack aims to inject malicious code into the product or software used by the target to gain access to the targeted system. Security the software supply chain has become priority for organizations and governments worldwide. Development of modern software relies on third party components, open-source libraries and automated pipelines. But it all comes at a cost of compromising with the software security as it opens doors for threat actors. The proposed solutions aim to provide an end-to-end protection and enhanced visibility into the supply chain ecosystem. The design and implementation of a software solution that continuously monitors, assesses, and mitigates risks throughout the software development lifecycle. By integrating automated Software Bill of Materials (SBOM) generation, vulnerability scanning, real-time threat intelligence, and anomaly detection powered by machine learning.**

***Keywords*—SBOM (Software bills of materials), Machine learning, threat intelligence, supply chain**

1.INTRODUCTION

Nowadays software development has shifted from isolated in-house production to a highly distributed model. They rely on external components, third-party services and automated workflows. This has reduced the development cost and raced the innovation. However, it comes with a prices of security vulnerabilities of the software supply chain. Sophisticated cyberattacks on supply chain software has been frequently increased in recent years. This has posed serious threats to national security, enterprises resilience, and public trust in technology systems.

Incidents such as SolarWinds orion breach in 2020, log4j vulnerability (Log4Shell) discovered in late 2021. These incidents demonstrated how threat actors can infiltrate the trust of vendors to compromise software updates. These updates are distributed to thousands of clients including government agencies and critical infrastructure operators. These kinds of cases underscore a stark reality that even well maintained and trusted software can become vectors for large-scale, cascading cyber incidents.

The software supply chain encompasses every element that contributes to the production and delivery of a software product. This includes proprietary and open-source code, package dependencies, applications programming interfaces (APIs), build and integration tools, containerization platforms and infrastructure as code templates. Example of build and integration tools are Jenkins, GitHub actions, or GitLab CI/CD. They all together manages deployment environments. If they are not properly monitored and secured, each component can become a potential attack surface. Techniques such as dependency confusion, typo squatting, and malicious package injection can be used by attacks to compromise the supply chain. They can bypass the traditional perimeters defenses in the process.

This research paper aims to explore a robust solution for securing the software supply chain. Solution focuses on embedding security into development lifecycle itself. It shifts the security practices "left" into earlier stages of development and extending them "right" into the production monitoring. Through the integration of automated Software Bill of Materials (SBOM) generation, vulnerability scanning, machine learning-driven anomaly detection and real time threat intelligence. The system is designed to provide comprehensive visibility and active protection across all phases of software development and deployment. It offers a holistic approach to software supply chain security that aligns with emerging standards and industry best practices.

2. OBJECTIVES

The main objectives of the proposed solution are to provide a comprehensive, integrated and proactive security platform. The architecture of the solution address not just the vulnerabilities in isolated stages but throughout the entire software development lifecycle. The need for unified security approach becomes imperative as the dependency on open-source ecosystems and cloud native infrastructure increases.

A key component of this solution is the automated generation and validation of software bills of materials SBOM. The system integrates a real-time vulnerability scanning engine that continuously checks components and containers against authoritative databases and other CVE repositories. Databases such as the NVD (national vulnerability database), GitHub advisory database. This enables organizations to maintain complete visibility into their software components. They can trace the origin of all libraries or artifacts used in a build and identify high-risk dependencies.

The platform incorporates machine learning (ML) models trained to recognize anomalous behaviors in development and build patterns. These models detect any suspicious deviations. Any unexpected changes in dependency versions, unauthorized access to repositories or irregular build frequencies can be flagged by the model. These may indicate

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 04, April-2025

malicious activity or insider threats. By adopting behavioral analytics, the system can detect zero-day attacks and sophisticated supply chain compromises.

3. SYSTEM DESIGN AND ARCHITECTURE:

The software is built upon a modular and scalable architecture. It is designed to accommodate the needs of software development environment. The design emphasizes both extensibility and interoperability. The modularity ensures that each component functions independently yet cohesively, which enables the seamless integration of existing development workflows across various CI/CD pipelines, cloud platforms and version control systems.

At its core, the system includes a module for generating SBOMs in standard formats such as SPDX and CycloneDX. This is responsible for parsing build artifacts and recording component metadata. The SBOM serves as the foundational layer for vulnerability assessment, component lineage tracking and audit readiness.

Then comes the vulnerability scanner module, which cross-references the SBOM. It matches against the public and private vulnerability databases such as National Vulnerability Database (NVD), and vendor-specific feeds. By automating this cross-matching process, the system promptly identifies known CVES. This thereby enables timely mitigation actions.

To enhance detection, machine learning engine is integrated with the architecture. This learns from historical patterns and detects anomalies. The system is capable of flagging of inconsistencies such as unusual build durations, inexperienced dependency updates or changes in developer access patterns. These kinds of patterns may indicate a compromise to supply chain.

An embedded policy enforcement engine allows organizations to codify and automate security risks. Disallowing unsigned packages, blocking vulnerable components, or mandating dependency freshness are some risks that can be blocked using this engine. Integrated with threat intelligence that ingest real time data from reputable sources. This allows the solution to dynamically respond to threats and known indicators of compromise (IOCS).

Real-time threat intelligence, which is also an essential aspect of the system. It continuously ingests and analyzes contextual data from global security feeds. Zero-day alerts, newly disclosed vulnerabilities, attack indicators and emerging TTPS (tactics, techniques and procedures) are included.

A graphical user interface (GUI) developed using react for easy accessibility. The interface offers real-time visualizations. Users can easily navigate through dashboard to monitor their software supply chain. They can receive alerts, manage policy configurations and can see the SBOM lineage. Also reports can be generated for compliance and audit purposes.

The backend services are developed using python and go. These languages provide concurrency and ecosystem support for handling data-intensive and security focused tasks. Restful APIs are used to facilitate communication between microservices and external tools. Data persistence is managed through scalable databases such as PostgreSQL.

4. FLOWCHART

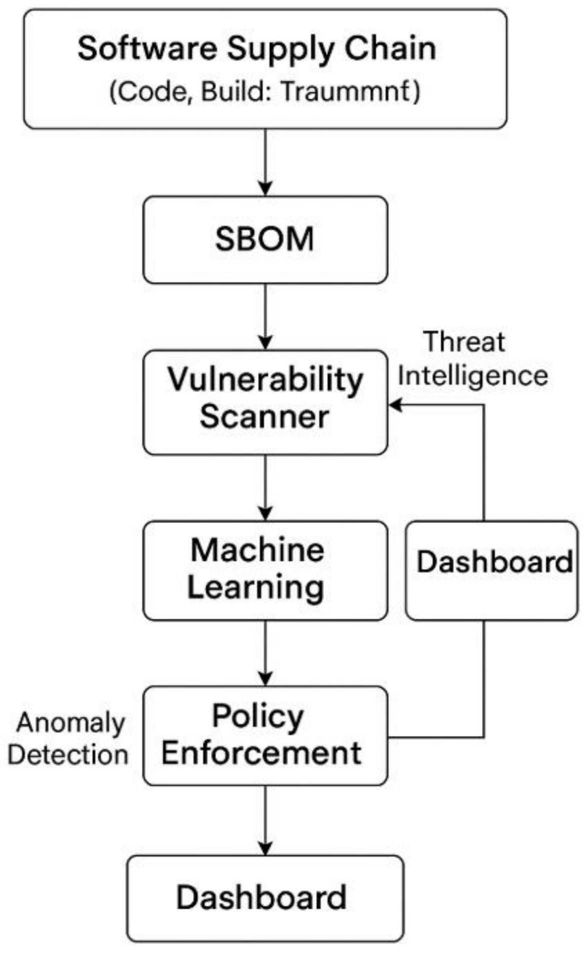

Figure 1: Workflow of proposed solution

The flowchart represents the architecture of the software. the process begins with the ingestion of source code and build artifacts from repositories. the inputs are passed to sbom generator, which extracts and formats the metadata. then it is analyzed by vulnerability scanner comparing component versions against public and private vulnerabilities databases. parallel to this works the machine learning engine to monitor behavior of the data and any defined attack patterns. issues are evaluated against the policy enforcement engine and if any policy is violated then alerts are triggered. threat intelligence advances the analysis by analyzing it with up to data IOCS and threat context. user can monitor the findings and activities using the centralized dashboard. alerts, track dependency trees and configurations can be managed using dashboard. the whole architecture offers robust end to end protection for the software development lifecycle. It combines modularity, automation and real-time responsiveness.

5. METHODOLOGY

The approach for designing the software supply chain begins with an in-depth examination of high-profile cyber incidents. SolarWinds and log4j vulnerability were studied to understand the failure points which allowed these attacks to propagate undetected. These case studies helped to identify the common patterns, weakness in the existing defenses.

Following this, an engineering phase was undertaken to ensure that proposed system effectively addresses gaps that were observed in traditional practices. All the functional and non- functional requirements were documented properly. All the features, SBOM generation, automated vulnerability scanning, machine learning engine, policy enforcement engine and threat intelligence were listed in the document.

The system was designed with a modular and API driven framework which can seamlessly be integrated with various ci/cd tools and DevOps workflows. It will ensure the flexibility and scalability of the system. Development lifecycle follows an agile methodology. It will allow the system to take continuous feedbacks so that rapid enhancement of the features can be made based on testing outcome and stakeholder's inputs.

For rigorous testing of the software abilities to detect and respond to threats, open-source projects with known vulnerabilities to simulate attack scenarios within controlled environment are utilized. Projects which have malicious dependencies, outdated packages and unauthorized artifacts changes are used for testing. Anomaly detection and policy enforcement mechanism will get fine-tuned by testing in these scenarios.

Hybrid analysis approach will make it better to adopt in conditions with more complex and previously unseen attack patterns. Static analysis to asses component integrity and licensing risks and dynamic analysis to detect unusual changes in the development or build environment. It was critical in enhancing the accuracy and reliability of the system across the software lifecycle.

Using a set of quantitative and qualitative metrics for evaluating the effectiveness and efficiency of the system is the final stage.

Key performance indicators include the vulnerability detection rate, the frequency of false positives, false negatives and response latency. These metrics are important to assess both technical robustness and practical deploy ability of the system within real world.

6. PROPOSED MODEL

The proposed software solution operates on a layered architecture, where each layer plays a crucial role in securing the software supply chain:

1. SBOM Generator Layer: This layer scans and extracts component metadata from build artifacts to produce SBOMs in CycloneDX or SPDX formats, offering traceability of software components as shown in Figure 2.

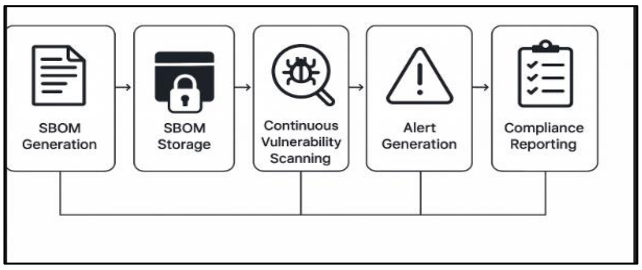

Figure 2: SBOM Lifecycle

2. Vulnerability Assessment Layer: This module compares component metadata with databases such as the NVD and GitHub Advisory Database, flagging outdated or vulnerable dependencies in real-time.
3. Anomaly Detection Layer: Using machine learning algorithms like Isolation Forest and Random Forest, this layer detects unusual behaviours in build and deployment pipelines, such as unexpected version changes or unauthorized modifications as shown in Figure 3.

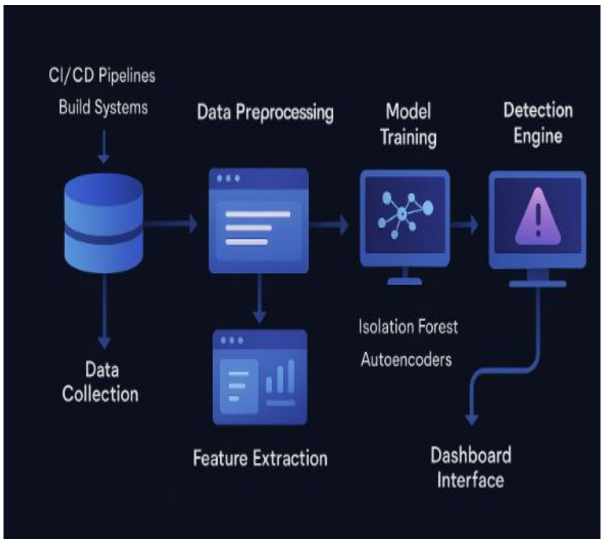

Figure 3: Machine learning model workflow

4. Policy Engine Layer: This rules-based module ensures compliance by enforcing organization-specific security policies—e.g., rejecting unsigned code or prohibiting deprecated libraries.
5. Threat Intelligence Integration Layer: Connects with external feeds like MISP or commercial intelligence services to update and act on the latest threats.
6. Visualization and Dashboard Layer: Displays dependency trees, real-time alerts, risk scores, and SBOM comparisons using an intuitive interface developed in React.

7. RESULT

The proposed solution was evaluated using open-source repositories which contain publicly disclosed vulnerabilities. These test environments were realistic so that software capabilities can be assessed properly. The system achieved high static vulnerability detection. It correctly identified approximately 92% of known issues as shown in Figure 4. And in parallel to that machine learning model showed an accuracy rate of 88%. It effectively flagged unusual and suspicious behaviours. False positives rate was 7.5% which was observed in anomaly detection. This value is within the range of acceptable limits for enterprise-level applications. It can be optimised through continuous training and feedback.

Performance metrics indicated that the system added less than 10% processing overhead to standard CI/CD pipeline execution times, thereby affirming its feasibility for integration into real-time development workflows. In the testing phase, intuitive nature of the system's alerting and dashboard interface features were highlighted by qualitative feedback from the developers. The non-intrusive design allowed for rapid adoption without disrupting ongoing development activities. These results collectively represent the effectiveness and reliability of the proposed solution. It showed the balance between the accuracy with usability and operational efficiency.

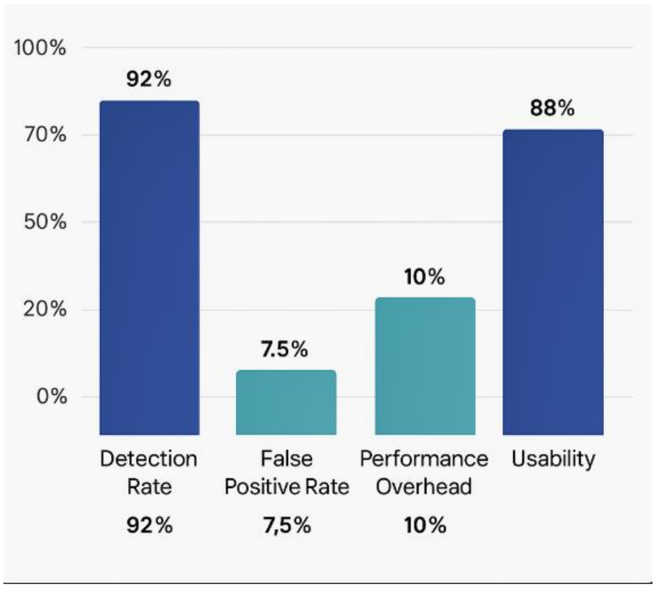

Figure 4: Result of the proposed model

8. CHALLENGES AND LIMITATIONS

While the proposed solution is effective and offers significant improvements, it also comes with challenges and limitations. One of the primary difficulties which the model has to deal with is the ever-evolving cyber threat landscape. It will require continuous updates to threat intelligence feed, vulnerability database, and detection algorithm to remain effective. The machine learning engine will require regular retraining and tuning to keep it up to date with new attack patterns. Technical challenges will also emerge when attempting to ensure seamless compatibility across the variety of ci/cd tools and version control systems. If the additional scanning and monitoring processes are not optimised it may hinder the speed and efficiency of the development pipelines. Organizational resistance can be one of the significant barriers. Developers and teams may be reluctant to adopt stricter security protocols due to concerns over increased complexity or reduced development agility. Overcoming these limitations requires a combination of technological refinement, user-centric design, and strategic change management efforts within organizations**.**

9**.** FUTURE WORK

To further strengthen the capabilities of the software several enhancements can be made. Blockchain can be one of the technologies to integrate it with. It will ensure the tamper-proof verification of software artifacts across the distributed environment. By leveraging the immutability and transparency features of blockchain, the system can provide verifiable proof of origin and integrity for each component involved in the supply chain. Furthermore, integration with DevSecOps tools and methodologies will allow security practices to be embedded within the software development lifecycle. Another advancement that can be made is use of knowledge graphs and graph-based anomaly detection techniques**.** This will enhance the system's ability to identify complex, multi-stage attacks. By understanding the relationships between different components and events within the ecosystem. Finally, expanding the platform's compatibility with a broader range of programming languages, build systems, and development environments will be a long-term goal. It will ensure that the solution remains versatile and accessible across various software development contexts.

10**.** CONCLUSION

Securing the software supply chain has become crucially important due to the increased complexity of software development and the sophistication of cyberattacks. A strong, comprehensive answer has been put out by this research to deal with the many hazards that crop up during the software development lifecycle. The platform offers a comprehensive approach to supply chain protection by integrating key security practices like automatic SBOM generation, ongoing vulnerability scanning, machine learning-driven anomaly detection, and integration with real-time threat intelligence feeds. The user-centric interface facilitates proactive monitoring and prompt threat response, while the modular and scalable design guarantees adaptability to a variety of development contexts. Ultimately, the proposed system empowers organizations to gain comprehensive visibility into their software components, identify and respond to threats early, and enforce consistent security policies across the pipeline. In doing so, it lays the groundwork for a more resilient and trustworthy software ecosystem capable of withstanding both present and emerging supply chain threats.

REFERENCES

[1] Zahan, N. (2023). Software supply chain risk assessment framework. 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 251–255. https://doi.org/10.1109/icse-companion58688.2023.00068
[2] N. Falliere, L. O. Murchu, and E. Chien, "W32.Stuxnet Dossier," Symantec, Security Response, 2011. [Online]. Available: https://www.symantec.com
[3] E. B. Kritzinger and S. H. von Solms, "Cyber security for home users: A new way of protection through awareness enforcement," Computers & Security, vol. 29, no. 8, pp. 840–847, 2010.
[4] S. Arora, S. Goel, A. H. Karp, R. Pandey, and D. R. Thomas, "Redundancy and diversity for defense of cyber-physical systems," IEEE Trans. Dependable Secure Comput., vol. 15, no. 5, pp. 799–810, Sept.-Oct. 2018.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 04, April-2025

[5] A. Bhandari, D. Tian, and V. Sekar, "Characterizing and detecting malicious third-party libraries in Android apps," in Proc. USENIX Security Symp., 2021, pp. 849–866.
[6] L. G. Reynaud and A. M. Malek, "Automated detection of vulnerable third-party components in mobile apps," in Proc. 40th Int. Conf. Softw. Eng. (ICSE), 2018, pp. 1074–1084.
[7] H. Okhravi, S. Bak, T. Hobson, D. Bigelow, and W. Streilein, "Data Flow Authentication in Software Supply Chain Security," ACM Trans. Priv. Secur., vol. 22, no. 4, pp. 1–26, 2019.
[8] L. Allodi and F. Massacci, "Security events and vulnerability data for cyber security risk estimation," Risk Analysis, vol. 37, no. 8, pp. 1606–1627, 2017.
[9] M. Souppaya and K. Scarfone, "Guide to Software Supply Chain Security," NIST SP 800-161r1, 2021.
[10] M. Pashenkov and N. G. Ivanov, "Approaches to Ensuring Security of the Software Supply Chain," in Proc. Int. Conf. Inf. Technol. Math. Mech. Opt. (ITMMO), 2020, pp. 512–516.
[11] D. L. Parnas, "Software Aging," in Proc. 16th Int. Conf. Softw. Eng., Sorrento, Italy, 1994, pp. 279–287.
[12] S. Rahman, J. King, M. Nabeel, "SBOMs in Software Supply Chain Security: A Practical Overview," IEEE Security & Privacy, vol. 20, no. 2, pp. 65–72, 2022.
[13] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the Common Vulnerability Scoring System version 3.1," FIRST.org, 2019.
[14] J. Bau and J. Mitchell, "Security modeling and analysis," IEEE Security & Privacy, vol. 9, no. 3, pp. 18–25, 2011.
[15] S. Newman, Building Microservices: Designing Fine-Grained Systems, O'Reilly Media, 2015.
[16] OWASP, "OWASP Dependency-Check," [Online]. Available: https://owasp.org/www-project-dependency-check/
[17] GitHub, "GitHub Advisory Database," [Online]. Available: https://github.com/advisories
[18] CISA, "Defending Against Software Supply Chain Attacks," 2022. [Online]. Available: https://www.cisa.gov
[19] Google, "SLSA: Supply-chain Levels for Software Artifacts," 2022. [Online]. Available: https://slsa.dev
[20] SPDX, "Software Package Data Exchange (SPDX) Specification," 2023. [Online]. Available: https://spdx.dev
[21] CycloneDX, "CycloneDX SBOM Standard," OWASP Foundation, 2023. [Online]. Available: https://cyclonedx.org.
[22] Gunda, S. K. (2025). Accelerating Scientific Discovery With Machine Learning and HPC-Based Simulations. In Integrating Machine Learning Into HPC-Based Simulations and Analytics (pp. 229-252). IGI Global Scientific Publishing.
[23] Gunda, S. K. (2024, September). Analyzing Machine Learning Techniques for Software Defect Prediction: A Comprehensive Performance Comparison. In 2024 Asian Conference on Intelligent Technologies (ACOIT) (pp. 1-5). IEEE.