Connecting SBOM and CVE: An Insightful Study of Software Vulnerabilities

Immanuel Anthony*, PLS Jayalaxmi*, Rahul Saha†$, Gulshan Kumar†$, Mauro Conti†
* Department of Computer Science, Bhavan's Vivekananda College, Telengana, India
$School of Computer Science and Engineering, Lovely Professional University, Punjab, India
† Department of Mathematics, University of Padua, Padua, Italy

Abstract—Analyzing vulnerabilities in software supply chains is critical for mitigating security risks. The Software Bill of Materials (SBOM) provides transparency by listing all components and dependencies. Additional sources like the National Vulnerability Database (NVD), Open Web Application Security Project (OWASP), and the Common Vulnerabilities and Exposures (CVE) system enhance vulnerability tracking. However, relying on a single source leads to incomplete analyses, as vulnerabilities in dependencies may still pose threats even when the primary software appears secure. Therefore, we require a comprehensive process to provide information on dependency vulnerability and an understanding of general software vulnerabilities.

To address the above-mentioned problems, we introduce SOMVE (SbOM and cVE), a process based on Machine Learning (ML) model to integrate SBOM and CVE. This integration provides comprehensive vulnerability information, which is not available by SBOM or CVE individually otherwise. As such SBOM provides 5 metrics and CVE contains 21 metrics. Therefore, we identify and extract the matching CVIDs that are common between both datasets. The Random Forest (RF) model used in SOMVE traces the vulnerabilities (using CVIDs) missed by focusing exclusively on the SBOM data. Specifically, SOMVE includes key details such as vulnerability scores, impacts, descriptions, and assessments. A rigorous experiment on SOMVE shows 97% accuracy in mapping the CVE IDs from both sources. Finally, SOMVE generates a consolidated JSON report with 11 metrics that highlight the importance of comprehensive vulnerability analysis to secure software supply chains.

Index Terms—Software Bill of Materials (SBOM), Vulnerability Analysis, Common Vulnerabilities, Random Forest Model.

I. Introduction

Managing complex software systems and addressing vulnerabilities is the most challenging task for security in software supply chain systems. The dependency on third-party libraries and open-source components often introduces hidden security risks. Moreover, the interconnected nature of these components complicates manual vulnerability tracking across software stacks. Thus, effective methods are needed to detect and mitigate vulnerabilities to prevent exploitation [1]. The Software Bill of Materials (SBOM) and Common Vulnerabilities and Exposures (CVE) offer detailed insights into components and their associated risks. In contrast to NVD and OWASP, SBOM ensures transparency in dependencies, whereas CVE provides precise vulnerability tracking. By combining these two sources, a combined analysis that covers more gaps than is possible with other sources alone. We detail SBOM and CVE's importance in the background work I-A and outline the paper's contribution and organization in the subsequent subsections.

A. Background

The SBOM serves as a fundamental tool to enhance software supply chain security. This providess a comprehensive inventory of all software components used in a product and offers a clear view of its dependencies [2]. This transformation enables the organization to monitor and identify vulnerabilities by linking SBOM data with vulnerability data. Despite their importance, SBOMs face several limitations, such as inconsistent formats, difficulty identifying components, and dependency resolution issues, resulting in incomplete and inaccurate SBOMs that reduce their effectiveness in ensuring security [3]. Advanced techniques with integrated models enhance vulnerability detection by analyzing relationships between software components and uncovering hidden vulnerabilities. These approaches provide deeper insight into the risks associated with dependencies. Integrating SBOM generation into pipelines enables continuous monitoring and updates, ensuring proactive security [4]. However, to fully realize the potential of SBOMs in securing software supply chains, it is essential to address their inherent limitations, such as format inconsistencies and challenges in dependency resolution. Due to these issues, SBOMs often fail to provide complete and accurate insights into software vulnerabilities. A more structured and automated approach is required to overcome these challenges. With the integration of SBOMs with vulnerability databases, organizations can identify and mitigate security risks more efficiently and more precisely, enabling them to address potential threats. Addressing these limitations and providing a consolidated report to identify the vulnerabilities is the main aim of our study.

B. Contribution

This study integrates SBOM with CVE metadata to improve vulnerability detection in software supply chains. Its major contributions are given below.

- SBOM-CVE Integration The combination of SBOM with CVE metadata is achieved by matching CVE IDs using the $RandomForestalgorithm$. SBOM contributes four key metrics: $Severity$, $Score$, $Method$, and $Vector$, while CVE not only supports these four but also provides seven additional metrics. This combined approach significantly enhances the detection and analysis of vulnerabilities and addresses these gaps that would remain undetected by relying on either SBOM or CVE independently.
- Vulnerability Mapping Combining SBOM and CVE metrics into a consolidated JSON file enables vulnerability detection by linking component dependencies with specific threat data. This approach improves the accuracy of threat identification, reduces false positives, and accelerates response times. In the future, such an integration will improve over automated security assessments and continuous vulnerability monitoring, required for real-time threat management in evolving software ecosystems.

C. Paper organization

The detailed structure of this paper is organized as follows. Section II provides a comprehensive review of the relevant literature, highlighting the latest models and methodologies that leverage SBOMs for vulnerability analysis. Section I-B highlights the unique contributions of our work, emphasizing its novelty and significance in addressing the identified challenges. Section III outlines the proposed methodology, encompassing dataset selection, data preprocessing, and the implementation of machine learning models. Furthermore, it describes the experimental setup and evaluation criteria. Section IV presents the experimental results, offering a comparative analysis of various experiments conducted. Section V highlights the key findings and provides a comparison with existing studies. Finally, Section VI explores potential directions for future research aimed at improving security measures and enhancing vulnerability detection.

II. Literature Review

The Software Bill of Materials strengthens software supply chain security by providing a detailed inventory of components and dependencies essential to software creation. However, sharing SBOMs presents several challenges, including the risk of data tampering and the hesitation of software providers to fully disclose detailed information. These challenges limit the broader adoption and effective use of SBOMs, underscoring the need for more secure and adaptable sharing mechanisms. Some of the latest studies having an impact on SBOM data, and integrated with machine learning models to improve security are discussed below:

Recent research offers a diverse look at SBOM's role in securing software supply chains and handling the challenges right from adoption to innovation across various contexts. Axelsson et al. [5] reviewed SBOM adoption in open-source software, while Adewumi et al. [6] examined quality assessment models to refine software development approaches. Camp 's [7] findings support SBOM as a means to strengthen both supply chain security and component tracking. In a novel approach by Xia et al. [8]. Extended SBOM to AI systems with a blockchain-enabled "Artificial Intelligence Bill of Materials" (AIBOM) a framework for secure sharing. Ding et al. [9] proposed an enterprise-focused SBOM generation method to improve data precision, and Kemppainen [10] underscored practical considerations in the selection of SBOM. Furthermore, Chaora et al. and [11] emphasized SBOM's impact on risk management across industrial sectors. Additionally, Harer et al. [12]. Added a proactive layer to software security by introducing a machine-learning approach to automate vulnerability discovery. Collectively, these studies highlight SBOM's vital role in protecting software ecosystems and its expanding applications. The methods used by all of the models listed above to analyze SBOM data are similar. Yet, every model has its own set of drawbacks and provides distinct insights. Table I describes the gaps found in each model.

III. SOMVE-Methodology

In this paper, we propose a model to enhance vulnerability management by integrating data from SBOM files with CVE records from CVE.org [13]. Our model utilizes Random Forest a ML algorithm, to cross-reference and analyze these datasets based on CVID. The detailed procedure is given in Figure 1.

Model Workflow: The SOMVE provides a structured approach, as shown in Figure 1, to integrate and analyze data from SBOM files and CVE records

TABLE I
Findings and Gaps in Existing SBOM Studies

Reference (Citation)	Findings and Gaps
Axelsson et al. [5]	Studied SBOM adoption in open-source software but lacked focus on proprietary ecosystems and secure sharing mechanisms.
Adewumi et al. [6]	Proposed quality models for software development but did not address SBOM quality's impact on vulnerability analysis.
Camp [7]	Focused on SBOM for supply chain security but lacked real-time vulnerability tracking in dynamic environments.
Xia et al. [8]	Introduced AIBOM using blockchain for AI systems but missed broader applicability to general software ecosystems.
Ding et al. [9]	Focused on precise SBOM generation for enterprises but did not consider scalability or adaptability in diverse supply chains.
Kemppainen [10]	Discussed SBOM tool selection but ignored challenges in enterprise-wide integration.
Chaora et al. [11]	Highlighted SBOM in industrial risk management but overlooked cross-sector integration and scalability.
Harer et al. [12]	Used ML for vulnerability detection but relied on static analysis and lacked SBOM integration or real-time updates.

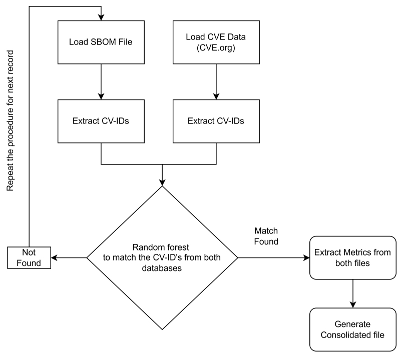
Load SBOM File
Load CVE Data (CVE.org)
Extract CV-IDs
Extract CV-IDs
Repeat the procedure for next record
Random forest to match the CV-ID's from both databases
Not Found
Match Found
Extract Metrics from both files
Generate Consolidated file

Fig. 1. Working procedure of the SOMVE

for improved vulnerability management. The process starts with loading the required data from both sources, followed by extracting and cross-referencing CVE identifiers (CV-IDs) using a Random Forest algorithm. This approach helps to identify the vulnerabilities linked to specific software components with consolidated information generated. The key stages of the model are explained below: Loading Data: We start by loading the SBOM file and the corresponding CVE data from CVE.org [13]. Extracting CV-IDs: From both datasets, we extract CVE identifiers (CV-IDs). These IDs are unique references to specific vulnerabilities. Matching CV-IDs Using Random Forest: Our model uses a Random Forest algorithm to compare and match the extracted CV-IDs from the SBOM and CVE data. This step ensures the accurate identification of vulnerabilities associated with software components. Handling Results: If a match is found, we proceed to extract relevant metrics from both the SBOM and CVE data, such as severity scores, impact metrics, and remediation steps. These metrics are then compiled into a consolidated JSON file, providing a comprehensive view of the identified vulnerabilities. Iterative Process: In case no match is found, the model iterates to process the next record. This ensures the thorough analysis of all entries in the datasets. Output:The final output is a consolidated JSON file that integrates key insights from both data sources. This file aids in decision-making processes by providing detailed vulnerability metrics and recommendations.

A. Dataset

To address the problems discussed in the literature review Section II, we introduce SOMVE (SbOM and cVE), a ML based process to integrate SBOM and CVE. The primary goal of this study is to integrate the SBOM data [14] with the CVE IDs available in the CVE dataset [13] and generate a consolidated report to track vulnerabilities. SBOM files provide detailed information, including the name, version, licensing details, supplier information, and a dependency graph of each component. Additionally, SBOMs include features such as checksum and hash values to verify the integrity of components, along with metadata like build environments and timestamps. In this study, we extracted SBOM files using publicly available GitHub data sources [14]. The extracted metrics (5/5 from SBOM) are formatted into JSON files with selected key metrics listed in Table II. Furthermore, we utilized the dataset provided by cve.org [13], maintained through the CVE List repository on GitHub. This dataset includes Common Vulnerabilities and Exposures (CVE) records, for identifying and analyzing software vulnerabilities. We have considered 11 metrics out of 21 Provided by the CVE dataset. Finally, compressed JSON files are generated and matched with individual CVE(IDs) and metadata. The additional metrics extracted from the CVE dataset are projected in Table II.

TABLE II
Metrics available in SBOM and CVE

SBOMFields	CVE Fields
CVE-IDs	CVE-IDs
Attack-vector	Attack-description
Base-score	Attack-complexity
Method	User-interaction
Base-severity	Confidentiality-impact
	Integrity-impact
	Privileges required
	Vector string

Integrating SBOM data with the CVE records given in Table II enables risk assessment by linking the details of the software components (e.g. attack description, base score, complexity, dependency, and interaction) with known vulnerabilities (e.g. CVE IDs, severity metrics, affected products). This integration provides comprehensive vulnerability information, which is not available by SBOM or CVE individually otherwise. Specifically, SOMVE includes key details such as vulnerability scores, impacts, descriptions, and assessments.

B. Data Preprocessing

As part of our preprocessing methodology, we used python scripts to restructure both datasets, creating new JSON files that retained only the required metrics while discarding the repeated ones. We have considered four key metrics along with the corresponding CVE IDs provided in the SBOM and excluded certain metrics from the CVE dataset to streamline the requirements, as shown in Table ??.

a) Standardization of Objects:: As part of preprocessing, key objects are renamed and mapped to ensure a consistent structure across the dataset. Examples of these transformations include:

- cve.values.metrics is renamed to metrics.
- containers.adp.metrics is mapped to cveMetadata.
- containers.cna.descriptions is standardized as descriptions.

This renaming and mapping process ensures the dataset maintains a uniform structure, enabling seamless analysis and model training.

b) Final data structure:: After preprocessing, each JSON file contains only three standardized objects:

- cveMetadata: Contains metadata of the CVE file.
- Descriptions: Includes detailed descriptions of the vulnerabilities.
- Metrics: Provides CVE-related metrics, such as severity scores and impact details.

This streamlined data structure enhances efficiency in analysis and facilitates subsequent model training and evaluation.

C. Experiment

Algorithm 1 begins by loading the datasets, which consist of input features from both SBOM and CVE files and their corresponding target labels CVE-ID to match the software component from the CVE. We use Random Forest (RF) a machine learning technique to create an ensemble structure and aggregate the extractions of CVE IDs. RF is highly effective in handling mixed feature types with minimal input values. In this experiment, we work with various data types, including numerical values (base score), text (description), lists (attack vector), and date-time values (published date), to identify vulnerabilities. Compared to other Artificial Intelligence models, RF requires less parameter tuning, and results in effective interpretations. In our context, we used RF to simplify the task of matching CVE IDs from the CVE and SBOM datasets, but not for prediction. The ability to identify patterns and correlations between the features of both datasets made RF an ideal choice for this problem. Resulting in accurate and reliable matches without extensive training.

We start with significant pre-processing in the dataset to establish consistency and uniformity across all records as the first step in Algorithm 1. The original dataset consists of numerous JSON files, each containing multiple objects. As the next step, we scan these JSON files and extract the three essential objects: cveMetadata, descriptions, and metrics. These objects are critical as they provide the necessary information for further analysis. During this processing, we systematically scan each JSON file to locate the sub-objects within cveMetadata, descriptions, and metrics. Once a relevant sub-object is identified, we extract the entire object. This process is repeated for all three key objects in the data set. To ensure uniformity, we standardize the object names throughout the JSON files. Given that the original JSON files contain varying structure names, we map these to their corresponding standardized names.

IV. Results and Findings

The outcome of our experiment is to evaluate a consolidated JSON file based on the matches extracted from the CVE IDs in both datasets. This experiment highlights the importance of preprocessing and standardizing the dataset, and ensures the model operates efficiently and consistently without encountering structural discrepancies. Our main aim is to retrieve relevant data associated with a given

Algorithm 1 CVE and SBOM Consolidation Using Random Forest

```
1: Input: CVE Dataset $D_{CVE}$, SBOM Dataset $D_{SBOM}$
2: Output: Consolidated JSON Report $R_{JSON}$
3: Step 1: Load Datasets
4: $D_{CVE} \leftarrow$ Load dataset from CVE (JSON files)
5: $D_{SBOM} \leftarrow$ Load dataset from SBOM
6: Step 2: Preprocess CVE Dataset
7: for each JSON file $f \in D_{CVE}$ do
8:     Extract essential objects: cveMetadata, descriptions, and metrics.
9:     Map object names to standardized names:
10:         cve.values.metrics $\rightarrow$ metrics
11:         containers.adp.metrics $\rightarrow$ cveMetadata
12:         containers.cna.descriptions $\rightarrow$ descriptions
13:     Retain only the standardized objects in $f$.
14: end for
15: Step 3: Preprocess SBOM Dataset
16: $D'_{SBOM} \leftarrow D_{SBOM} -$ Meta-information
17: $F_{SBOM} \leftarrow$ Select top 5 features from $D'_{SBOM}$
18: Step 4: Run RF for Matching CVE IDs
19: Model $\leftarrow$ Train RF to match CVE IDs from $D_{CVE}$ and $F_{SBOM}$
20: Step 5: Match CV-IDs
21: for $id \in D_{CVE}$ do
22:     Match $\leftarrow$ Model.predict$(id, F_{SBOM})$
23: end for
24: Step 6: Generate JSON Report
25: $R_{JSON} \leftarrow$ {Matched IDs with relevant features}
26: Step 7: Save Report
27: Save $R_{JSON}$ to the specified path.
```

CVE ID. By working with standardized objects—cveMetadata, descriptions, and other metrics to trace severity. This model achieves 97% accuracy in matching CVE IDs to their corresponding JSON records.

SBOM Dataset and New Values: : Table III compares the metrics selected from the SBOM dataset and the new metrics added to the consolidated JSON file after processing with the RF model:

Table III presents a comparative analysis of the metrics derived from the SBOM dataset and the refined metrics introduced after processing through the Random Forest (RF) model, achieving 97% accuracy in matching CVE-IDs. The CVE dataset initially contained several redundant or irrelevant fields, such as Assigner Org ID, Assigner Short Name, Date published, Date updated, version, Timestamp, and State. These fields were excluded as they do not contribute directly to vulnerability analysis. Additionally, the vulnerabilities field in the SBOM dataset provided only minimal information, such as severity ratings, vector, and method, which limited the ability to conduct a detailed risk assessment.

TABLE III
Consolidated Metrics selected and fields eliminated from CVE- Dataset

Consolidated Metrics	Fields eliminated
CVE-IDs	Assigner Org ID
Attack-vector	Assigner Short Name
Base-score	Date Published
Attack Description	Date Updated
Base-severity	State
Attack complexity	CVSS Version
User-Interaction	Date Reserved
Confidentiality impact	Availability Impact
Integrity-impact	Scope
Privileges Required	Timestamp
Vector String	

To address these gaps, the dataset was enhanced with critical metrics selected from the CVE dataset. The selection of 11 metrics as: CVE-IDs, attackVector, baseScore, baseSeverity, attackDescription, attackComplexity, userInteraction, confidentialityImpact, integrityImpact, privilegesRequired, and vectorString. These retained metrics are selected based on the importance in assessing the severity, impact over vulnerability analysis. In contrast, fields like assignerOrgID, assignerShortName, datePublished, state, CVSS version, scope, timestamp, and others were excluded due to their administrative or secondary nature, which added noise without contributing directly to vulnerability detection.

The consolidated metrics now offer a comprehensive and actionable dataset for machine learning models, allowing for detailed risk assessments, improved prioritization of vulnerabilities, and informed mitigation strategies. This refinement ensures that organizations can achieve better precision and reliability in their vulnerability analysis processes.

V. Comparison

Both datasets considered for this study have redundant and irrelevant metrics, which results in larger file sizes and slower processing. Tracing the vulnerability with limited inputs and depending on a single data source resulted in inconsistent and hindered structural representation. The consolidated dataset generated by the RF model in this study resolves the issues by standardizing and enriching the data availability, enhancing accuracy and performance. We highlight our contribution and introduce a scalable SBOM generation method using machine learning to improve precision and coverage across diverse supply chains compared to existing models discussed in Table IV.

TABLE IV
Comparison of SBOM Studies with Proposed Model

Reference	Findings and Gaps	Our Contribution
Axelsson et al. [5]	Explored SBOM adoption in open-source software, which lacks secure sharing methods.	Open source JSON file to address trust and data integrity issues.
Adewumi et al. [6]	Quality assessment, lacks in vulnerability analysis.	Additional matrix provided to develop a prediction model.
Camp [7]	Focus on supply chain security.	Extend inputs to assess the risk.
Xia et al. [8]	AIBOM using blockchain, limited to analyze the ecosystems.	Apply blockchain-enabled techniques on SOMVE to avoid data tampering.
Ding et al. [9]	Focused on enterprise-specific SBOM.	Standardized structure to coverage across supply chains.
Kemppainen [10]	Practical exposure to SBOM, lacks in highlighting the challenges.	Highlighted the requirements supported with an integrated solution.
Chaora et al. [11]	Risk management assessment.	Present a cross-sector framework that is adaptable for any product.
Harer et al. [12]	Focus on static methods for vulnerability analysis	SOMVE ensure proactive security measures with risk-based metrics.

Existing SBOM studies highlight advancements but reveal significant gaps. Axelsson et al. [5] explored SBOM adoption in open-source software but lacked a framework for secure sharing in proprietary ecosystems, which our model addresses by ensuring trust and data integrity by adding additional fields. Adewumi et al. [6] proposed quality assessment models but didn't link SBOM quality to vulnerability analysis, which we address with an integrated SBOM quality with CVE dataset, which can be extended for vulnerability prediction. Camp [7] emphasized SBOM's role in supply chain security but overlooked real-time vulnerability tracking, which our approach resolves with proactive risk management with additional information required to train any classification models and handle real-time vulnerabilities. Xia et al. [8] proposed AIBOM for AI systems but couldn't generalize it to broader ecosystems; Our study can be extended using blockchain-enabled SBOM sharing for all software ecosystems. Ding et al. [9] focused on enterprise SBOM generation but lacked scalability, addressed by our machine learning-driven, scalable SBOM generation method. Kemppainen [10] considered SBOM tool selection but ignored integration challenges in diverse environments, resolved by our integration framework. Chaora et al. [11] emphasized SBOM for industrial risk management but neglected cross-sector applicability. Harer et al. [12] introduced machine learning for vulnerability detection but relied on static analysis without real-time updates or SBOM integration, these gaps are filled by integrating the security model on our JSON file for proactive security.

VI. Conclusion and Further Work

The presented study introduces SOMVE, a novel machine learning-based framework that integrates SBOM and CVE datasets to enhance software vulnerability analysis. Unlike prior models, SOMVE uses Random Forest to correlate CVE IDs across sources, generating a unified JSON report with 11 key metrics. This significantly improves vulnerability traceability and detection accuracy (97%). The consolidation addresses gaps left by standalone sources and enables proactive threat mitigation. In the future, we will extend SOMVE with real-time data feeds and predictive analytics to support dynamic vulnerability classification and cross-sector security applications.

VII. Acknowledgement

This work was supported by the European Commission under the Horizon Europe Programme, as part of the project LAZARUS (https://lazarus-he.eu /) (Grant Agreement no. 101070303). The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

References

[1] Hughes, C., & Turner, T. (2023). Software Transparency: Supply Chain Security in an Era of a Software-Driven Society. John Wiley & Sons.

[2] Xia, B., Bi, T., Xing, Z., Lu, Q., & Zhu, L. (2023, May). An empirical study on a software bill of materials: Where we stand and the road ahead. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 2630-2642). IEEE.

[3] Bi, T., Xia, B., Xing, Z., Lu, Q., & Zhu, L. (2024). On the way to some Investigating design issues and solutions in practice. ACM Transactions on Software Engineering and Methodology, 33(6), 1-25.

[4] Bjørntvedt, N. K., & Sæther, K. (2024). Designing the CI/CD Security Maturity Model (Master's thesis, University of Agder).

[5] Axelsson, V., & Larsson, F. (2023). Understanding the Software Bill Of Material for supply-chain management in Open Source projects.

[6] A. Adewumi, S. Misra, and N. Omoregbe, “Evaluating open source software quality models against iso 25010,” in 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. IEEE, 2015, pp. 872–877

[7] L. J. Camp and V. Andalibi, “Sbom vulnerability assessment & corresponding requirements,” NTIA Response to Notice and Request for Comments on Software Bill of Materials Elements and Considerations, 2021.

[8] B. Xia, D. Zhang, Y. Liu, Q. Lu, Z. Xing, and L. Zhu, “Trust in software supply chains: Blockchain-enabled sbom and the aibom future,” arXiv preprint arXiv:2307.02088, 2023.

[9] X. Ding, F. Zhao, L. Yan, and X. Shao, “The method of building boom based on enterprise big data,” in 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE). IEEE, 2019, pp. 1224– 1228

[10] P. Kemppainen, “Managing 3rd party software components with software bill of materials,” 2023.

[11] A. Chaora, N. Ensmenger, and L. J. Camp, “Discourse, challenges, and prospects around the adoption and dissemination of software bills of materials (bombs),” in 2023 IEEE International Symposium on Technology and Society (ISTAS). IEEE, 2023, pp. 1–4.

[12] J.Harer, J.A., Kim, L.Y., Russell, R.L., Ozdemir, O., Kosta, L.R., Rangamani, A., Hamilton, L.H., Centeno, G.I., Key, J.R., Ellingwood, P.M. and Antelman, E., 2018. Automated software vulnerability detection with machine learning. arXiv preprint arXiv:1803.04497.

[13] https://www.cve.org/Downloads

[14] https://github.com/CycloneDX/cyclonedx-cli

[15] Yu M, Zhuge J, Cao M, Shi Z, Jiang L. A Survey of Security Vulnerability Analysis, Discovery, Detection, and Mitigation on IoT Devices. Future Internet. 2020; 12(2):27. https://doi.org/10.3390/fi12020027