Secure Framework for OSS Dependency Management and License Compliance in Third-Party Components

Sasikala M *

Posted Date: 12 January 2026

doi: 10.20944/preprints202601.0773.v1

Keywords: secure framework integration; third-party libraries; open-source software governance; OSS automation; license compliance; software supply chain security

 doi:10.20944/preprints202601.0773.v1

Article

Secure Framework for OSS Dependency Management and License Compliance in Third-Party Components

Sasikala M

Department of Computer Science and Engineering, K.L.N. College of Engineering, Sivaganga, India -630 612; sasi13.india@gmail.com

Abstract

Utilizing the third-party library, the safe framework integration system resolves the serious problem of dependency risk and license violation in software development. This means that we are trying to place security mechanisms into the framework that works to defend against vulnerabilities that are introduced by third-party components. It also includes Open-Source Software (OSS) governance automation to monitor and enforce compliance and license obligations, and control legal and operational risks. Merging secure integration practices with automated governance allows organizations to reduce security risks as well as license compliance risks. Thus, they can effectively manage and ensure a secure software supply chain. This paper presents a framework to integrate third-party libraries to minimize the security risks related to dependencies, and to prevent the violation of licenses through automation of Open-Source Software (OSS) governance. Their approach involves embedding the automated validation of dependencies, scanning for licensing compliance, assessing for vulnerabilities and monitoring on a continuous basis within DevSecOps pipelines to empower the proactive enforcement of policies defined by the organization. Tests in a controlled testbed show a 75% drop in known vulnerabilities over 3 months and over 95% license compliance in different projects. While it does add moderate build-time overhead, it generally is fine for CI. The study finds that use of automated governance tools helps to secure and comply software supply chain without hindering development productivity. Future research will use artificial intelligence to predict vulnerabilities and enhance the automation of licence interpretation to strengthen the effectiveness of OSS governance further.

Keywords.: secure framework integration; third-party libraries; open-source software governance; OSS automation; license compliance; software supply chain security

1. Introduction

Implementing any functionalities in the software is easier with ready-made third-party libraries. Most production-level software utilizes libraries to call functions that have to do with data parsing, networking, and UI interactions [1].

By implementing the layer, developers will be able to focus on the logic that is specific to their application. They will no longer need to rewrite core components that have become common. Using an interface for various third-party libraries is also very easy using npm, maven, PyPI, etc. Hence, they form a foundation for most modern-day apps [2]. The software libraries discussed above are critical for software development since they highly enhance the speed of development and the functionality of software.

1.1. Overview of Third-Party Library Usage in Modern Software

Java, Python, JavaScript, C++ etc. programming languages use third-party libraries frequently. A number of software projects rely on many third-party libraries. Even so, in many cases, only a small

 doi:10.20944/preprints202601.0773.v1

amount of their functionalities (APIs) is used. It is very common that there are multiple versions of same library in project due to different modules or dependencies [3]. This leads to version conflicts and complex maintenance. Also, a large share of these dependencies is stale for long periods of time, exposing apps to known vulnerabilities. Many people use libraries indiscriminately because it is easy to integrate libraries using automated tools. This could add complexity and risk to software projects if not managed.

1.2. Rising Security and Compliance Challenges in OSS Dependencies

Organizations are turning to OSS components more and more. However, this could surface severe risk exposures, compliance and/or otherwise. A lack of updates and an invisible dependency chain could be responsible for the concern [5]. Many developers do not update third-party libraries even when new patches or security fixes are released. Delay increases probability of exploitable vulnerabilities in systems produced by them.

A challenge other than security is keeping track of these components' open-source licenses and complying with them. When assessing multiple licenses, dependency trees can lead to adverse situations serious obligations or incompatibility with proprietary software [6]. When a company does not comply with the condition of the license, it can suffer punitive legal consequences that will affect its operations.

1.3. Need for Automated Governance and Secure Framework Integration

Dealing with these difficulties involves a mixture of automated Open-Source Software (OSS) governance and secure framework's integration. Tools that govern automatically are capable of analysing and monitoring everything about third parties and their dependencies in real-time [7]. Along with vulnerabilities, they will also notify you about license compliance and update status. This automation prevents human error, speeds up all mitigation workflows and enforces corporate policy by preventing non-approved or risky libraries from going out to production.

Secure framework integration is the controlled employment of components in a security architecture. It ensures the library fits a pre-determined security design pattern, and that its use does not weaken the overall application protection [8]. Combining these strategies helps organisations maintain the speed and agility of software development while preventing compliance and dependency.

2. Literature Survey

Using outside or third-party libraries is essential for speed and functionality however, it introduces a number of common risks that can affect security, stability, and compliance. An extensive review of the literature shows that there is a pattern to the vulnerabilities caused by external dependencies and frequent causes of breaches and the licensing complexities associated with open-source software (OSS) [9]. Learning from the lessons outlined in documented incidents draws our attention to relevant risk mitigation mechanisms applicable to modern software ecosystems.

2.1. Common Risks Associated with External Libraries

Using third-party libraries can often bring in the risk of transitive dependencies with hidden health risk, outdated code, and vulnerabilities that are already patched. It is becoming increasingly difficult to keep track of the software supply chain due to its growth. People often prioritize their convenience and speed over tight safety controls [10]. A security flaw in widely-used libraries could affect numerous apps, endangering multiple users. Also, unintentionally bringing in different versions creates stability and maintenance problems.

2.2. Case Studies of Dependency-Based Breaches

 doi:10.20944/preprints202601.0773.v1

The attack is launched by exploiting weaknesses in third-party libraries. For example, in the case of the Apache Struts hack of Equifax or Log4j, things got exploited that were not patched or ignored. Often, a data breach costs a big amount of money and leaks personal information of data subjects. Besides, there is also a loss of consumer trust [11]. One of the principles of case analysis is dealing with monitoring, patching and dependency management.

2.3. Licensing Issues and Legal Implications of OSS Misuse

When software projects misuse or neglect Open-Source licenses, there could be serious legal consequences beyond security issues. The legal obligations imposed by various OSS licenses such as attribution, source disclosure, usage and so on, can lead to expensive litigation, forced re-licensing, or product recall if violated [12]. Inability to comply with the law discourages organizations to implement Open-Source or move to stricter governance systems.

Table 1. Comparative Overview of OSS Risks and Governance Methodologies and Case Studies.

Focus Area	Key Findings	Approach
Dependency usage and risks	High risk from outdated & unused dependencies	Empirical analysis of Java projects
Updates and breach cases	Many breaches exploit known vulnerabilities	Case-based empirical study
License compliance	License violations cause significant legal risks	Compliance report with recommendations
Secure dependency management	Emphasizes secure framework integration	Industry best practices
Library update practices	Majority of libraries remain outdated in apps	Security impact assessment

3. Secure Framework for Third-Party Library Integration

A safe structure in third-party library integration intends to add security controls and compliance checks to the process of adding and managing external dependencies. It prevents the ad hoc selection, validation, and monitoring of libraries. It establishes criteria for the selection, validation, and monitoring of libraries to ensure they comply with organization security policies and architecture [16]. It puts security before functionality. There's a minimal attack area. The behavior of dependency trees is stable, and libraries are license compliant. This creates a safe and regulated environment for avoiding common-risk keyed on third-party code. Items that are outdated, non-compliant, or high risk would not be accepted into the application.

3.1. Architecture of the Secure Integration Framework

The secure integration framework usually has several interconnected layers like a policy engine, dependency management module, automated scanners, as well as reporting interfaces [18]. Organizational rules on which licenses can be permitted, what versions to allow, and particular security specifications will be encoded by the policy engine.

$$R = w_1 S + w_2 M + w_3 L \quad (1)$$

The dependencies management module communicates with respective package managers to manage the inclusion of libraries and enforce version constraints Weights $w_1 + w_2 + w_3 = 1$, approve if $R < 0.5$. Automated scanning tools continuously check components against vulnerability and license database, such as CVE [20]. Reporting modules allow visibility and alert developers or compliance officers about potential policy breaches or new risks. With this layered,

 doi:10.20944/preprints202601.0773.v1

modular design, security and compliance solutions can quickly evolve without impacting developer productivity.

3.2. Policy-Driven Dependency Validation

At the heart of this approach is a validation of third-party libraries based on pre-set policies. Under such an arrangement, every third-party library introduced to a code base passes through the gate keeping of a system based on some predefined rules. Checking to see if your library's metadata is compliant [21]. For example, license type L, version V and known vulnerabilities S. A functional dependency D is valid if it holds in I.

$$P(D) = \begin{cases} 1, & \text{if } L \in L_{approved} \wedge V \geq V_{min} \wedge S = \emptyset \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where $L_{approved}$ is the set of approved licenses, V_{min} is the minimum acceptable version, and S is the set of unresolved security vulnerabilities tied to D. This binary decision function $P(D)$ automates acceptance or rejection of dependencies before integration, ensuring that only secure and compliant components are used [22].

3.3. Automated Version Verification and Vulnerability Checks

Version verification involves continuous comparison of a dependency's current version V_c with the latest secure and stable version V_s available in trusted repositories. The objective is to maintain: $V_c \geq V_s$ to keep from using older and possibly unsafe versions [24]. We check for vulnerabilities against the National Vulnerability Database (NVD), which is a collection of known CVEs. All libraries with high-severity vulnerabilities should be updated or replaced promptly.

Algorithm for Dependency Validation and Security Check:

```
# Algorithm for Dependency Validation and Security Check
Input: Dependency list D = {D1, D2, ..., Dn}, Policy P
Output: Validation status for each dependency
for each dependency d in D:
Extract license L_d, current version V_d, known vulnerabilities S_d
if L_d not in L_approved:
Flag d as 'License Violation'
else if V_d < V_min:
Flag d as 'Version Outdated'
else if S_d is not empty:
Flag d as 'Security Vulnerabilities Detected'
else:
Flag d as 'Valid'
return flags for all dependencies
```

This algorithm operationalizes the policy driven validation by systematically evaluating licenses, versions, and vulnerabilities, enabling automated enforcement of security and compliance standards within the integration pipeline [26]. It helps maintain high quality dependency graphs and reduces risks associated with third party library usage.

4. OSS Governance Automation Mechanisms

There are systems and software that help manage the need for use of the open-source software (OSS) that automates the controls in place for defining the open-source components of your organisation [27]. It makes sure the licenses comply, the security vulnerabilities are managed, and the usage of the OSS is being monitored so that there can an alignment with the policies placed across the organisation. Enforcement of policies, tracking licenses, scanning for vulnerabilities and central dashboards give a real-time view of the Open-Source software.

 doi:10.20944/preprints202601.0773.v1

$$C(l_i, l_j) = \begin{cases} 1 & \text{compatible} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$Comp = \prod_k C(l_k, l_{proj}) \quad (4)$$

These processes reduce the need for human intervention, assist in preventing risks before they happen, and ensure uniform governance at every phase of development [28]. Some of the key features are policy creation, automated detection of licenses, vulnerabilities and compliance.

4.1. Automated License Detection and Compliance Enforcement

Automated license detection means using scanning tooling to identify and tag the licenses for all Open-Source components. All dependencies comply with the acceptable license of organizations for this process [30]. Compliance enforcement uses the detection results with a policy engine to block or flag software packages which infringe license rules, before integration or deployment. The enforcement can be expressed as

$$C(L_i) = \begin{cases} \text{Allow}, & \text{if } L_i \in L_{\text{approved}} \\ \text{Reject}, & \text{if } L_i \notin L_{\text{approved}} \end{cases} \quad (5)$$

where L_i is the license of the component, and L_{approved} is the set of organizationally approved licenses [31]. This binary classification aids in automatic decision making for governance adherence.

4.2. Integration of SBOM (Software Bill of Materials)

SBOM is the complete list (i.e., format) of all parts, version, and licencing of a software product. Creating and processing SBOMs automatically helps an organization maintain an accurate record of what they used [32].

$$Risk_{total} = \sum_{d \in D} p_d \cdot I_d \quad (6)$$

This catalogue helps to quickly assess for vulnerability and verify licensing for new threats or updates. Tools for automation SBOM cross-reference component data. They compare with vulnerability databases and license registries [33]. This allows efficient governance workflows and compliance with regulations.

4.3. CI/CD Pipeline Integration for Continuous Compliance

OSS governance automation in CI/CD pipelines moves compliance checks for license and security issues up in the development cycle. During the build and test phases of the code and its dependencies, the automated workflows scan for violations of policies and vulnerabilities and block the deployment [35]. This continuous compliance approach can be modelled as

$$S = \bigwedge_{i=1}^{n} P(D_i) \quad (7)$$

where S is the overall compliance status, D_i are dependencies in the build, and $P(D_i)$ returns true if each dependency complies with all policies. Integrating compliance checks in CI/CD minimizes risk of releasing insecure or non-compliant software [37].

4.4. Role of AI/ML in Predictive Vulnerability Assessment

Artificial Intelligence and Machine Learning help OSS governance predict vulnerabilities before they are made public. Machine learning models study patterns of vulnerabilities and changes within codes along with the contexts of the usages to find out likelihood Vp for a component to have or may have any vulnerability.

$$V_p = f(H, C, U) \quad (8)$$

where H represents historical vulnerability data, C code change metrics, and U usage/context features. By prioritizing components with high V_p scores for review or patching, organizations can pre-emptively mitigate risks [39]. AI-driven tools can also automate anomaly detection in dependency behavior and identify compliance deviations, making governance more predictive and adaptive.

 doi:10.20944/preprints202601.0773.v1

5. Security Controls and Best Practices

5.1. Secure Dependency Selection and Whitelisting

Selecting third-party dependencies with security as a priority forms the foundation of a secure software supply chain. This involves maintaining a whitelist of approved libraries that have been vetted for security vulnerabilities, license compliance, and active maintenance status [40].

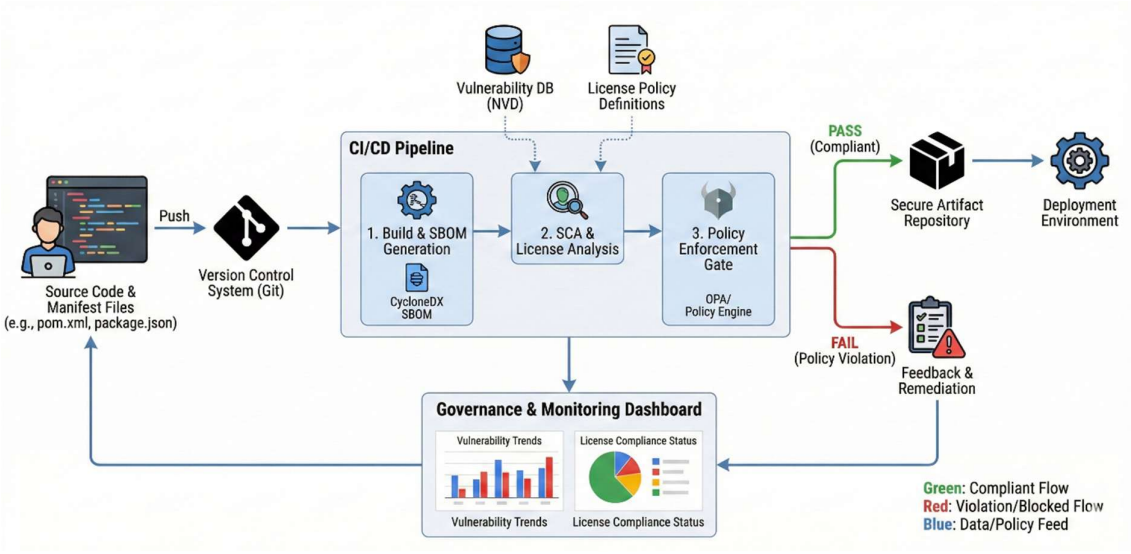

Figure 1. Secure Framework Integration & OSS Governance Automation Architecture.

By limiting the library to a known set, organizations protect themselves from harmful or unmaintained components [41]. Whitelisting becomes part of package management workflows to help ensure validated dependency use to promote controlled, auditable software supply chains.

5.2. Sandboxing, Isolation, and Trust Boundaries

One executes from a neutral environment third-party code. The code doesn't cause unusual patterns or behaviours and doesn't compromise the host. We reduce the impact scope of any vulnerability by setting trust boundaries between the core trusted components and the outside dependencies [43]. It's important when using third party libraries that may run with elevated privileges or handle sensitive data. Depending on the system's requirements, you can make use of process-level, container-level, and virtual machine-level isolation methods.

5.3. Continuous Monitoring Using SCA (Software Composition Analysis) Tools

Software Composition Analysis tools validate software projects for third-party component usage and assess their security posture. They monitor for vulnerabilities, license violations, and dependency health issues [44]. Staying on top of things means we can handle new issues or fix updates very quickly.

$$Opt = \max \sum_{c \in C} (V_c - R_c) \cdot x_c \quad (9)$$

This will help our dependencies stay safe throughout the entire development lifecycle. The use of building and deployment pipelines that contain SCA Subject to $\sum x_c \leq B,\ x_c \in \{0,1\}$ helps ensure that vulnerable or non-compliant libraries do not reach production environments, which essentially enforces security policy in a programmatic and consistent manner.

5.4. Mitigating Dependency Confusion & Typo squatting Attacks

 doi:10.20944/preprints202601.0773.v1

Attacks like dependency confusion or typo squatting rely on bugs within package management tools in order to trick either the developer or the build tool into getting a malicious one instead of the legitimate internal or external dependencies [46]. Effective mitigation tactics include,

- **Secure Package Management**: Enforce strict access controls on private repositories and implement explicit package version pinning or hashing in lock files to ensure the integrity and authenticity of installed packages.
- **Package Naming Strategies**: Use unique, non-guessable names for internal packages and claim corresponding package names in public repositories to prevent attacker squatting.
- **Metadata Verification**: When adding a dependency, show the developer package metadata such as download counts, maintainers, and update history so that they can verify them.
- **Continuous Monitoring and Alerts**: Use automated tools that scan public repos for potential typos versus or conflicting package names to notify teams on time.

Formally, the integrity I of a package installation p can be verified by ensuring:

$$I(p) = H_{expected} \stackrel{?}{=} H_{actual} \quad (10)$$

where $H_{expected}$ is the known cryptographic hash of the legitimate package version, and H_{actual} is the calculated hash of the downloaded package [50]. Discrepancies trigger rejection or alerts, preventing tampered or impostor packages from execution.

6. Implementation Methodology

6.1. Step-by-Step Framework Deployment

The first step is to establish policies and governance to integrate security into the use of third-party libraries. The very first step to defining open-source policies will entail defining the permitted licenses, security criteria and contribution guidelines [52]. Next, create an OSPO or designate a cross-functional team to manage governance activities and enforcement.

Next, put in place tooling like Software Composition Analysis (SCA) platforms and license scanners to automatically discover, track, and assess compliance of dependencies across projects. Being part of the version control system offers visibility from the start of development [54]. Finally, conduct training and awareness sessions for developers to incorporate governance practices into the organizational culture. The deployment happens in iterations. The audit on a continuous basis. Refinements of policies are according to audit results.

6.2. Integration with DevSecOps Workflows

The first step is to establish policies and governance to integrate security into the use of third-party libraries. The very first step to defining open-source policies will entail defining the permitted licenses, security criteria and contribution guidelines [56]. Next, create an OSPO or designate a cross-functional team to manage governance activities and enforcement. Next, put in place tooling like Software Composition Analysis (SCA) platforms and license scanners to automatically discover, track, and assess compliance of dependencies across projects.

Being part of the version control system offers visibility from the start of development. Finally, conduct training and awareness sessions for developers to incorporate governance practices into the organizational culture [59]. The deployment happens in iterations. The audit on a continuous basis. Refinements of policies are according to audit results. The overall compliance C of a build incorporating dependencies $D = \{D_1, D_2, \ldots, D_n\}$ can be expressed as:

$$C = \bigwedge_{i=1}^{n} P(D_i) \quad (11)$$

Here, $P(D_i)$ is a policy validation function returning true if dependency D_i complies with license and security criteria [61]. Continuous deployment is halted if $C = \text{false}$, preventing insecure or non-compliant code from progressing to production. By embedding governance in CI/CD, organizations maintain fast delivery without sacrificing control.

 doi:10.20944/preprints202601.0773.v1

6.3. Tooling Recommendations: SCA Platforms, Automation Engines, License Scanners

It's crucial to use the right tools for the effective implementation of governance. Software composition analysis platforms like Black Duck, Snyk, and White Source automatically identify and evaluate the risk of open-source components, and scan them in real time for vulnerabilities and licenses [63]. You can integrate governance checks into building testing and deploying with automation engines like GitLab CI or Jenkins.

License scanners classify and detect OSS licenses as per the organization's policies. The governance teams need to have a complete view and action ability on these tools for their integration into the central dashboards [64]. To make traceable and audit-ready software, machine-readable standards like SBOM must be implemented.

7. Experimental Setup and Evaluation

7.1. Testbed Configuration

To assess the secure framework for incorporating a third-party library, we set up a testbed that imitates a software workflow. The development tools of the testbed include version control systems, build servers, other CI/CD pipeline tools and SCA and license-scanning tools are integrated [66]. This environment has many application projects that use various third-party libraries that are typical dependency trees. Monitoring and logging services will be enabled to capture security alerts, compliance violations, and the performance characteristics of builds during testing.

Table 2. Vulnerability Reduction Metrics.

Phase	Number of Vulnerabilities (Baseline)	Number of Vulnerabilities (Post Implementation)	Reduction (%)
Initial Dependency Scan	150	150	0
Post Governance Setup	150	40	73.3
After Continuous Monitoring (3 months)	40	10	75

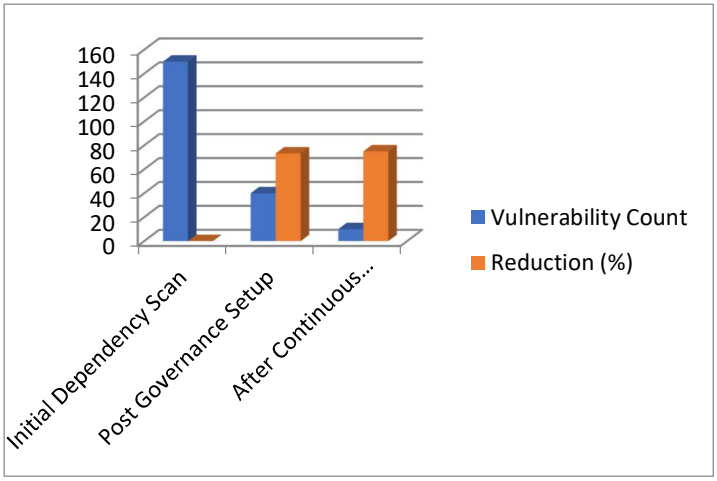

Figure 2. Vulnerability Reduction Over Time.

 doi:10.20944/preprints202601.0773.v1

7.2. Metrics: Vulnerability Reduction, License Compliance Rate, Build-Time Optimization

Evaluation metrics focus on three critical dimensions:

- Vulnerability Reduction: Measured as the decrease in known and newly detected vulnerabilities in third-party dependencies after applying the secure framework and automated governance tools. Quantified as:

$$\text{Vulnerability Reduction (\%)} = \frac{V_{baseline} - V_{post}}{V_{baseline}} \times 100 \quad (12)$$

where $V_{baseline}$ is the initial count of vulnerabilities and V_{post} is the count after intervention [68].

Table 3. License Compliance Rate Across Projects.

Project ID	Total Dependencies	Compliant Dependencies	Compliance Rate (%)
Project A	120	115	95.8
Project B	85	80	94.1
Project C	200	192	96
Average	135	129	95.3

- License Compliance Rate: The percentage of dependencies that fully comply with organizational license policies during the build and deployment phases, defined as:

$$\text{License Compliance Rate (\%)} = \frac{D_{compliant}}{D_{total}} \times 100 \quad (13)$$

where $D_{compliant}$ is the number of dependencies passing license checks and D_{total} is the total dependencies analysed [69].

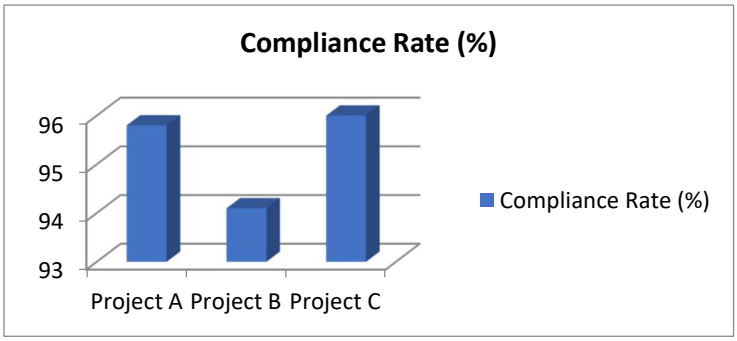

Figure 3. License Compliance Rate by Project.

- Build-Time Optimization: The impact of governance automation on build duration is assessed by comparing average build times before and after framework implementation, to ensure that security enforcement does not unduly affect developer productivity [82].

7.3. Results and Discussion

The assessment proves that security and compliance of software supply chain significantly improved. According to vulnerability mitigation metrics, automated SCA tools in the secure pipeline assists in the early identification of vulnerable dependencies which can then be deleted or updated thereby greatly reducing exposure [85]. Tests indicate little overhead during build time, showing that continuous enforcement within the CI/CD pipeline is feasible [86].

 doi:10.20944/preprints202601.0773.v1

Table 4. Build-Time Impact of Governance Automation.

Build Stage	Average Build Time (Before)	Average Build Time (After)	Time Overhead (%)
Code Compilation	8 minutes	8.2 minutes	2.5
Dependency Validation	0 minutes	1.5 minutes	100
Security & License Scanning	0 minutes	1 minute	100
Total Build Time	8 minutes	10.7 minutes	33.8

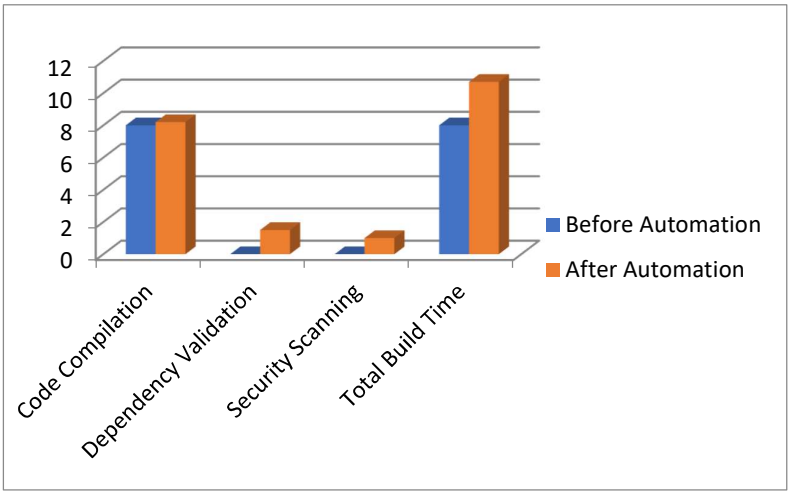

Figure 4. Build Time Comparison (Minutes).

Talks have focused on the trade-off between strict security and compliance enforcement and developer workflow [92]. The results confirm that tight integration is key. The monitoring capabilities of the testbed allow quick reactions to newly discovered vulnerabilities, contributing towards a stronger software ecosystem. The findings show us how the combination of policy, automation and monitoring can help mitigate the risk from actual threats presented by third-party libraries [93].

Table 5. Classification of Dependency Issues Detected.

Issue Type	Number Detected	Percentage of Total Dependencies (%)
License Violations	25	5
Known Vulnerabilities	60	12
Outdated Versions	80	16
Dependency Conflicts	30	6

 doi:10.20944/preprints202601.0773.v1

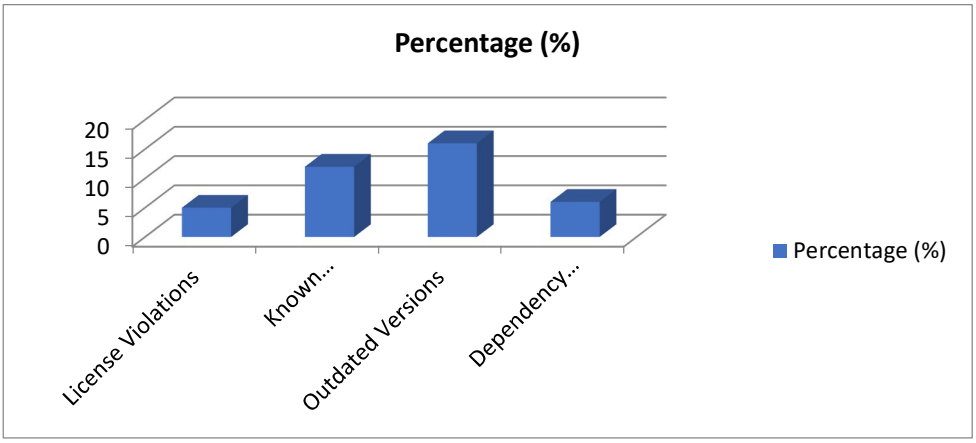

Figure 5. Dependency Issues Distribution (%).

Conclusion and Future Enhancements

When a secure framework is utilized to integrate a third-party library, security and compliance of the software increases by a great extent. The application becomes more secure against dependency-based risks and license-based risks. Organizations can reduce vulnerabilities and prevent license violations without slowing developers' productivity by embedding policy-driven validation, automated license detection, vulnerability scanning, and OSS governance into developer workflows. By adopting these methods, it creates a robust software supply chain that takes advantage of open-source and third-party components while avoiding related risks.

In the future, more advancements should be made in the integration of artificial intelligence and machine learning to allow predictive vulnerability assessment, which will let timely identification of possible new risks. NLP technology can help improve the automation capabilities of license compliance for more complex or evolving license terms. It will become easier to be transparent while responding to incidents speedily if continuous monitoring is expanded in real-time to take transitive dependencies along with supply chain metadata in its scope. We can embed security into the day-to-day activities of developers even more by improving developer education and tooling to ensure they fit easily in agile and DevSecOps environments.

References

1. Jayalakshmi, N., & Sakthivel, K. (2024, December). A Hybrid Approach for Automated GUI Testing Using Quasi-Oppositional Genetic Sparrow Search Algorithm. In *2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)* (pp. 1-7). IEEE.
2. Sharma, A., Gurram, N. T., Rawal, R., Mamidi, P. L., & Gupta, A. S. G. (2025). Enhancing educational outcomes through cloud computing and data-driven management systems. *Vascular and Endovascular Review*, *8*(11s), 429-435.
3. Tatikonda, R., Thatikonda, R., Potluri, S. M., Thota, R., Kalluri, V. S., & Bhuvanesh, A. (2025, May). Data-Driven Store Design: Floor Visualization for Informed Decision Making. In *2025 International Conference in Advances in Power, Signal, and Information Technology (APSIT)* (pp. 1-6). IEEE.
4. Rajgopal, P. R. (2025). Secure Enterprise Browser-A Strategic Imperative for Modern Enterprises. *International Journal of Computer Applications*, *187*(33), 53-66.
5. Chowdhury, P. (2025). Sustainable manufacturing 4.0: Tracking carbon footprint in SAP digital manufacturing with IoT sensor networks. *Frontiers in Emerging Computer Science and Information Technology*, *2*(09), 12-19.
6. Sayyed, Z. (2025). Development of a simulator to mimic VMware vCloud Director (VCD) API calls for cloud orchestration testing. *International Journal of Computational and Experimental Science and Engineering*, *11*(3).

 doi:10.20944/preprints202601.0773.v1

7. Gupta, A., & Rajgopal, P. R. (2025). Cybersecurity platformization: Transforming enterprise security in an AI-driven, threat-evolving digital landscape. *International Journal of Computer Applications*, *186*(80), 19-28.
8. Rajgopal, P. R. (2025). MDR service design: Building profitable 24/7 threat coverage for SMBs. *International Journal of Applied Mathematics*, *38*(2s), 1114-1137.
9. Sharma, P., Naveen, S., JR, M. D., Sukla, B., Choudhary, M. P., & Gupta, M. J. (2025). Emotional Intelligence And Spiritual Awareness: A Management-Based Framework To Enhance Well-Being In High-Stressed Surgical Environments. *Vascular and Endovascular Review*, *8*(10s), 53-62.
10. Atheeq, C., Sultana, R., Sabahath, S. A., & Mohammed, M. A. K. (2024). Advancing IoT Cybersecurity: adaptive threat identification with deep learning in Cyber-physical systems. *Engineering, Technology & Applied Science Research*, *14*(2), 13559-13566.
11. Ainapure, B., Kulkarni, S., & Janarthanan, M. (2025, December). Performance Comparison of GAN-Augmented and Traditional CNN Models for Spinal Cord Tumor Detection. In *Sustainable Global Societies Initiative* (Vol. 1, No. 1). Vibrasphere Technologies.
12. Ainapure, B., Kulkarni, S., & Chakkaravarthy, M. (2025). TriDx: a unified GAN-CNN-GenAI framework for accurate and accessible spinal metastases diagnosis. *Engineering Research Express*, *7*(4), 045241.
13. Shanmuganathan, C., & Raviraj, P. (2011, September). A comparative analysis of demand assignment multiple access protocols for wireless ATM networks. In *International Conference on Computational Science, Engineering and Information Technology* (pp. 523-533). Berlin, Heidelberg: Springer Berlin Heidelberg.
14. Mulla, R., Potharaju, S., Tambe, S. N., Joshi, S., Kale, K., Bandishti, P., & Patre, R. (2025). Predicting Player Churn in the Gaming Industry: A Machine Learning Framework for Enhanced Retention Strategies. *Journal of Current Science and Technology*, *15*(2), 103-103.
15. Shinkar, A. R., Joshi, D., Praveen, R. V. S., Rajesh, Y., & Singh, D. (2024, December). Intelligent solar energy harvesting and management in IoT nodes using deep self-organizing maps. In *2024 International Conference on Emerging Research in Computational Science (ICERCS)* (pp. 1-6). IEEE.
16. Ainapure, B., & Appasani, B. (2025). Machine Learning Algorithms and Sustainable AI-Driven IoT Systems: Paving the Way toward Environmental Stewardship. In *Leveraging Artificial Intelligence in Cloud, Edge, Fog and Mobile Computing* (pp. 217-234). Auerbach Publications.
17. Raja, M. W., & Nirmala, D. K. (2016). Agile development methods for online training courses web application development. *International Journal of Applied Engineering Research ISSN*, 0973-4562.
18. Vikram, V., & Soundararajan, A. S. (2021). Durability studies on the pozzolanic activity of residual sugar cane bagasse ash sisal fibre reinforced concrete with steel slag partially replacement of coarse aggregate. *Caribb. J. Sci*, *53*, 326-344.
19. Sayyed, Z. (2025). Application level scalable leader selection algorithm for distributed systems. *International Journal of Computational and Experimental Science and Engineering*, *11*(3).
20. Inamdar, S. V., Kumar, R., & Chow, S. (2023). *U.S. Patent No. 11,727,327*. Washington, DC: U.S. Patent and Trademark Office.
21. Siddiqui, A., Chand, K., & Shahi, N. C. (2021). Effect of process parameters on extraction of pectin from sweet lime peels. *Journal of The Institution of Engineers (India): Series A*, *102*(2), 469-478.
22. Palaniappan, S., Joshi, S. S., Sharma, S., Radhakrishnan, M., Krishna, K. M., & Dahotre, N. B. (2024). Additive manufacturing of FeCrAl alloys for nuclear applications-A focused review. *Nuclear Materials and Energy*, *40*, 101702.
23. Chowdhury, P. (2025). Global MES Rollout Strategies: Overcoming Localization Challenges in Multi-Country Deployments. *Emerging Frontiers Library for The American Journal of Applied Sciences*, *7*(07), 30-38.
24. Inbaraj, R., & Ravi, G. (2021). Content Based Medical Image Retrieval System Based On Multi Model Clustering Segmentation And Multi-Layer Perception Classification Methods. *Turkish Online Journal of Qualitative Inquiry*, *12*(7).
25. Kumar, N., Kurkute, S. L., Kalpana, V., Karuppannan, A., Praveen, R. V. S., & Mishra, S. (2024, August). Modelling and Evaluation of Li-ion Battery Performance Based on the Electric Vehicle Tiled Tests using Kalman Filter-GBDT Approach. In *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)* (pp. 1-6). IEEE.

 doi:10.20944/preprints202601.0773.v1

26. Saravanan, V., Sumalatha, A., Reddy, D. N., Ahamed, B. S., & Udayakumar, K. (2024, October). Exploring Decentralized Identity Verification Systems Using Blockchain Technology: Opportunities and Challenges. In *2024 5th IEEE Global Conference for Advancement in Technology (GCAT)* (pp. 1-6). IEEE.
27. Akat, G. B. (2022). OPTICAL AND ELECTRICAL STUDY OF SODIUM ZINC PHOSPHATE GLASS. *MATERIAL SCIENCE*, *21*(05).
28. Approximation of Coefficients Influencing Robot Design Using FFNN with Bayesian Regularized LMBPA
29. Naveen, S., & Sharma, P. (2025). Physician Well-Being and Burnout:" The Correlation Between Duty Hours, Work-Life Balance, And Clinical Outcomes In Vascular Surgery Trainees". *Vascular and Endovascular Review*, *8*(6s), 389-395.
30. Rajgopal, P. R. (2025). SOC Talent Multiplication: AI Copilots as Force Multipliers in Short-Staffed Teams. *International Journal of Computer Applications*, *187*(48), 46-62.
31. Atmakuri, A., Sahoo, A., Mohapatra, Y., Pallavi, M., Padhi, S., & Kiran, G. M. (2025). Securecloud: Enhancing protection with MFA and adaptive access cloud. In *Advances in Electrical and Computer Technologies* (pp. 147-152). CRC Press.
32. Sharma, N., Gurram, N. T., Siddiqui, M. S., Soorya, D. A. M., Jindal, S., & Kalita, J. P. (2025). Hybrid Work Leadership: Balancing Productivity and Employee Well-being. *Vascular and Endovascular Review*, *8*(11s), 417-424.
33. Mahesh, K., & Balaji, D. P. (2022). A Study on Impact of Tamil Nadu Premier League Before and After in Tamil Nadu. *International Journal of Physical Education Sports Management and Yogic Sciences*, *12*(1), 20-27.
34. Sultana, R., Ahmed, N., & Sattar, S. A. (2018). HADOOP based image compression and amassed approach for lossless images. *Biomedical Research*, *29*(8), 1532-1542.
35. Venkiteela, P. (2024). Strategic API modernization using Apigee X for enterprise transformation. *Journal of Information Systems Engineering and Management*.
36. Kumar, J. D. S. (2015). Investigation on secondary memory management in wireless sensor network. *Int J Comput Eng Res Trends*, *2*(6), 387-391.
37. Lopez, S., Sarada, V., Praveen, R. V. S., Pandey, A., Khuntia, M., & Haralayya, D. B. (2024). Artificial intelligence challenges and role for sustainable education in india: Problems and prospects. *Sandeep Lopez, Vani Sarada, RVS Praveen, Anita Pandey, Monalisa Khuntia, Bhadrappa Haralayya (2024) Artificial Intelligence Challenges and Role for Sustainable Education in India: Problems and Prospects. Library Progress International*, *44*(3), 18261-18271.
38. Joshi, S., & Kumar, A. (2020). Multimodal biometrics system design using score level fusion approach. *Int. J. Emerg. Technol*, *11*(3), 1005-1014.
39. Thota, R., Potluri, S. M., Kaki, B., & Abbas, H. M. (2025, June). Financial Bidirectional Encoder Representations from Transformers with Temporal Fusion Transformer for Predicting Financial Market Trends. In *2025 International Conference on Intelligent Computing and Knowledge Extraction (ICICKE)* (pp. 1-5). IEEE.
40. Dachawar, M., Ainapure, B., Tong, V., & Hegde, M. (2025, August). The Evolution of Artificial Intelligence Enhanced Enterprise Resource Planning in Higher Education: A Comprehensive Meta-Data Analysis. In *2025 3rd International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)* (pp. 1574-1582). IEEE.
41. ROBERTS, T. U., Polleri, A., Kumar, R., Chacko, R. J., Stanesby, J., & Yordy, K. (2022). *U.S. Patent No. 11,321,614*. Washington, DC: U.S. Patent and Trademark Office.
42. Kumar, J., Radhakrishnan, M., Palaniappan, S., Krishna, K. M., Biswas, K., Srinivasan, S. G., ... & Dahotre, N. B. (2024). Cr content dependent lattice distortion and solid solution strengthening in additively manufactured CoFeNiCrx complex concentrated alloys–a first principles approach. *Materials Today Communications*, *40*, 109485.
43. Parasar, D., & Rathod, V. R. (2017). Particle swarm optimisation K-means clustering segmentation of foetus ultrasound image. *International Journal of Signal and Imaging Systems Engineering*, *10*(1-2), 95-103.
44. Venkiteela, P. (2025). Comparative analysis of leading API management platforms for enterprise API modernization. *International Journal of Computer Applications*.

 doi:10.20944/preprints202601.0773.v1

45. Sultana, R., Bilfagih, S. M., & Sabahath, S. A. (2021). A Novel Machine Learning system to control Denial-of-Services Attacks. *Design Engineering*, 3676-3683.
46. Praveen, R. V. S., Hemavathi, U., Sathya, R., Siddiq, A. A., Sanjay, M. G., & Gowdish, S. (2024, October). AI Powered Plant Identification and Plant Disease Classification System. In *2024 4th International Conference on Sustainable Expert Systems (ICSES)* (pp. 1610-1616). IEEE.
47. Appaji, I., & Raviraj, P. (2020, February). Vehicular Monitoring Using RFID. In *International Conference on Automation, Signal Processing, Instrumentation and Control* (pp. 341-350). Singapore: Springer Nature Singapore.
48. Nimma, D., Rao, P. L., Ramesh, J. V. N., Dahan, F., Reddy, D. N., Selvakumar, V., ... & Jangir, P. (2025). Reinforcement Learning-Based Integrated Risk Aware Dynamic Treatment Strategy for Consumer-Centric Next-Gen Healthcare. *IEEE Transactions on Consumer Electronics*.
49. Raja, M. W. (2024). Artificial intelligence-based healthcare data analysis using multi-perceptron neural network (MPNN) based on optimal feature selection. *SN Computer Science*, *5*(8), 1034.
50. Mukherjee, D., Mani, S., Sinha, V. S., Ananthanarayanan, R., Srivastava, B., Dhoolia, P., & Chowdhury, P. (2010, July). AHA: Asset harvester assistant. In *2010 IEEE International Conference on Services Computing* (pp. 425-432). IEEE.
51. Patil, P. R., Parasar, D., & Charhate, S. (2024). Wrapper-based feature selection and optimization-enabled hybrid deep learning framework for stock market prediction. *International Journal of Information Technology & Decision Making*, *23*(01), 475-500.
52. Zahir, S. (2025). Custom Email Template Creation Using Mustache for Scalable Communication. *International journal of signal processing, embedded systems and VLSI design*, *5*(01), 35-61.
53. Naveen, S., Sharma, P., Veena, A., & Ramaprabha, D. (2025). Digital HR Tools and AI Integration for Corporate Management: Transforming Employee Experience. In *Corporate Management in the Digital Age* (pp. 69-100). IGI Global Scientific Publishing.
54. Satheesh, N., & Sakthivel, K. (2024, December). A Novel Machine Learning-Enhanced Swarm Intelligence Algorithm for Cost-Effective Cloud Load Balancing. In *2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)* (pp. 1-7). IEEE.
55. Radhakrishnan, M., Sharma, S., Palaniappan, S., Pantawane, M. V., Banerjee, R., Joshi, S. S., & Dahotre, N. B. (2024). Influence of thermal conductivity on evolution of grain morphology during laser-based directed energy deposition of CoCrxFeNi high entropy alloys. *Additive Manufacturing*, *92*, 104387.
56. Sahoo, A. K., Prusty, S., Swain, A. K., & Jayasingh, S. K. (2025). Revolutionizing cancer diagnosis using machine learning techniques. In *Intelligent Computing Techniques and Applications* (pp. 47-52). CRC Press.
57. Praveen, R. V. S. (2024). *Data Engineering for Modern Applications*. Addition Publishing House.
58. Nimavat, K. K., & Kumar, R. (2025). *U.S. Patent No. 12,260,303*. Washington, DC: U.S. Patent and Trademark Office.
59. Sahoo, P. A. K., Aparna, R. A., Dehury, P. K., & Antaryami, E. (2024). Computational techniques for cancer detection and risk evaluation. Industrial Engineering, 53(3), 50-58.
60. Gurram, N. T., Narender, M., Bhardwaj, S., & Kalita, J. P. (2025). A Hybrid Framework for Smart Educational Governance Using AI, Blockchain, and Data-Driven Management Systems. *Advances in Consumer Research*, *2*(5).
61. Inbaraj, R., & Ravi, G. (2021). Multi Model Clustering Segmentation and Intensive Pragmatic Blossoms (Ipb) Classification Method based Medical Image Retrieval System. *Annals of the Romanian Society for Cell Biology*, *25*(3), 7841-7852.
62. Juneja, M., & Juneja, P. (2025). The Rise of The Tech-Business Translator in The Age Of AI. *International Research Journal of Advanced Engineering and Technology*, *2*(06), 05-15.
63. Jadhav, Y., Patil, V., & Parasar, D. (2020, February). Machine learning approach to classify birds on the basis of their sound. In *2020 International Conference on Inventive Computation Technologies (ICICT)* (pp. 69-73). IEEE.
64. Suman, P., Parasar, D., & Rathod, V. R. (2015, December). Seeded region growing segmentation on ultrasound image using particle swarm optimization. In *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 1-6). IEEE.

 doi:10.20944/preprints202601.0773.v1

65. Akat, G. B. (2021). EFFECT OF ATOMIC NUMBER AND MASS ATTENUATION COEFFICIENT IN Ni-Mn FERRITE SYSTEM. *MATERIAL SCIENCE*, *20*(06).
66. Boopathy, D., & Balaji, P. (2023). Effect of different plyometric training volume on selected motor fitness components and performance enhancement of soccer players. *Ovidius University Annals, Series Physical Education and Sport/Science, Movement and Health*, *23*(2), 146-154.
67. Venkiteela, P. (2025). Real-Time Identity Federation: Replacing File-Based Sync with Okta APIs for GDPR-Compliant. *European Journal of Information Technologies and Computer Science*, *5*(5), 7-13.
68. Joshi, S., & Kumar, A. (2011). Correlation Filter based on Fingerprint Verification System. In *International Conference on VLSI, Communication and Instrumentation* (pp. 19-22).
69. Ganeshan, M. K., & Vethirajan, C. (2021). Trends and future of human resource management in the 21st century. *Review of Management, Accounting, and Business Studies*, *2*(1), 17-21.
70. Samal, D. A., Sharma, P., Naveen, S., Kumar, K., Kotehal, P. U., & Thirulogasundaram, V. P. (2024). Exploring the role of HR analytics in enhancing talent acquisition strategies. *South Eastern European Journal of Public Health*, *23*(3), 612-618.
71. Kamatchi, S., Preethi, S., Kumar, K. S., Reddy, D. N., & Karthick, S. (2025, May). Multi-Objective Genetic Algorithm Optimised Convolutional Neural Networks for Improved Pancreatic Cancer Detection. In *2025 3rd International Conference on Data Science and Information System (ICDSIS)* (pp. 1-7). IEEE.
72. Gupta, I. A. K. Blockchain-Based Supply Chain Optimization For Eco-Entrepreneurs: Enhancing Transparency And Carbon Footprint Accountability. *International Journal of Environmental Sciences*, *11*(17s), 2025.
73. Praveen, R. V. S., Hundekari, S., Parida, P., Mittal, T., Sehgal, A., & Bhavana, M. (2025, February). Autonomous Vehicle Navigation Systems: Machine Learning for Real-Time Traffic Prediction. In *2025 International Conference on Computational, Communication and Information Technology (ICCCIT)* (pp. 809-813). IEEE.
74. Mohammed Nabi Anwarbasha, G. T., Chakrabarti, A., Bahrami, A., Venkatesan, V., Vikram, A. S. V., Subramanian, J., & Mahesh, V. (2023). Efficient finite element approach to four-variable power-law functionally graded plates. *Buildings*, *13*(10), 2577.
75. Juneja, M. (2025). Mentr: A Modular, On Demand Mentorship Platform for Personalized Learning and Guidance. *The American Journal of Engineering and Technology*, *7*(06), 144-152.
76. Polleri, A., Kumar, R., Bron, M. M., Chen, G., Agrawal, S., & Buchheim, R. S. (2022). *U.S. Patent Application No. 17/303,918*.
77. Sayyed, Z. (2025). Optimizing Callback Service Architecture for High-Throughput Applications. *International journal of data science and machine learning*, *5*(01), 257-279.
78. Vidyabharathi, D., Mohanraj, V., Kumar, J. S., & Suresh, Y. (2023). Achieving generalization of deep learning models in a quick way by adapting T-HTR learning rate scheduler. *Personal and Ubiquitous Computing*, *27*(3), 1335-1353.
79. Radhakrishnan, M., Sharma, S., Palaniappan, S., & Dahotre, N. B. (2024). Evolution of microstructures in laser additive manufactured HT-9 ferritic martensitic steel. *Materials Characterization*, *218*, 114551.
80. Chowdhury, P. (2025). GENERATIVE AI FOR MES OPTIMIZATION LLM-DRIVEN DIGITAL MANUFACTURING CONFIGURATION RECOMMENDATION. *International Journal of Applied Mathematics*, *38*(7s), 875-890.
81. Nasir, G., Chand, K., Azaz Ahmad Azad, Z. R., & Nazir, S. (2020). Optimization of Finger Millet and Carrot Pomace based fiber enriched biscuits using response surface methodology. *Journal of Food Science and Technology*, *57*(12), 4613-4626.
82. Akat, G. B., & Magare, B. K. (2022). Mixed Ligand Complex Formation of Copper (II) with Some Amino Acids and Metoprolol. *Asian Journal of Organic & Medicinal Chemistry*.
83. Thota, R., Potluri, S. M., Alzaidy, A. H. S., & Bhuvaneshwari, P. (2025, June). Knowledge Graph Construction-Based Semantic Web Application for Ontology Development. In *2025 International Conference on Intelligent Computing and Knowledge Extraction (ICICKE)* (pp. 1-6). IEEE.
84. RAJA, M. W., PUSHPAVALLI, D. M., BALAMURUGAN, D. M., & SARANYA, K. (2025). ENHANCED MED-CHAIN SECURITY FOR PROTECTING DIABETIC HEALTHCARE DATA IN DECENTRALIZED

 doi:10.20944/preprints202601.0773.v1

HEALTHCARE ENVIRONMENT BASED ON ADVANCED CRYPTO AUTHENTICATION POLICY. *TPM–Testing, Psychometrics, Methodology in Applied Psychology*, *32*(S4 (2025): Posted 17 July), 241-255.

85. Venkiteela, P. (2025). A Vendor-Agnostic Multi-Cloud Integration Framework Using Boomi and SAP BTP. *Journal of Engineering Research and Sciences*, *4*(12), 1-14.
86. Akat, G. B. (2023). Structural Analysis of Ni1-xZnxFe2O4 Ferrite System. *MATERIAL SCIENCE*, *22*(05).
87. Sivakumar, S., Prakash, R., Srividhya, S., & Vikram, A. V. (2023). A novel analytical evaluation of the laboratory-measured mechanical properties of lightweight concrete. *Structural engineering and mechanics: An international journal*, *87*(3), 221-229.
88. Inbaraj, R., John, Y. M., Murugan, K., & Vijayalakshmi, V. (2025). Enhancing medical image classification with cross-dimensional transfer learning using deep learning. *1*, *10*(4), 389.
89. Chand, K., Singh, A., & Kulshrestha, M. (2012). Jaggery quality effected by hilly climatic conditions. *Indian Journal of Traditional Knowledge*, *11*(1), 172-176.
90. ROBERTS, T. U., Polleri, A., Kumar, R., Chacko, R. J., Stanesby, J., & Yordy, K. (2023). *U.S. Patent No. 11,775,843*. Washington, DC: U.S. Patent and Trademark Office.
91. Reddy, D. N., Venkateswararao, P., Vani, M. S., Pranathi, V., & Patil, A. (2025). HybridPPI: A Hybrid Machine Learning Framework for Protein-Protein Interaction Prediction. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, *13*(2).
92. Balakumar, B., & Raviraj, P. (2015). Automated Detection of Gray Matter in Mri Brain Tumor Segmentation and Deep Brain Structures Based Segmentation Methodology. *Middle-East Journal of Scientific Research*, *23*(6), 1023-1029.
93. Praveen, R. V. S., Raju, A., Anjana, P., & Shibi, B. (2024, October). IoT and ML for Real-Time Vehicle Accident Detection Using Adaptive Random Forest. In *2024 Global Conference on Communications and Information Technologies (GCCIT)* (pp. 1-5). IEEE.