# Code Review Weekly Workshop

## Comprehensive Security & Dependency Management Review Document

## 1. Introduction & Workshop Context

This document captures the full scope of discussion from the Code Review Weekly Workshop. The primary focus was security within code review, with deep exploration into dependency management, supply chain risk, vulnerability disclosure processes, and architectural defense strategies. The discussion emphasized that code review extends beyond functional correctness into long-term system safety, maintainability, and organizational resilience.

## 2. Dependency Management in Code Review

Introducing a dependency during a Merge Request (MR) was identified as a high-impact architectural decision. A dependency may introduce thousands of lines of runtime code and multiple transitive dependencies. Reviewers must recognize that adding a dependency alters the attack surface, maintenance burden, and long-term system risk.

- Prefer the latest stable version of dependencies to avoid silent security patches.
- Evaluate maintenance activity: commit frequency, release cadence, and contributor diversity.
- Prefer narrowly scoped libraries with single responsibilities.
- Conduct explicit build-vs-buy analysis for small, stable functionality.
- Inspect full dependency trees for transitive risk exposure.
- Discuss all concerns openly within the Merge Request discussion.

## 3. Supply Chain & CI/CD Risk

Modern dependency risk extends into CI/CD (Continuous Integration / Continuous Deployment) environments. Malicious packages may activate conditionally within CI pipelines, targeting secrets or API keys. Dependency introduction must be treated as a supply chain security decision.

## 4. CVE & Vulnerability Disclosure

CVE (Common Vulnerabilities and Exposures) identifiers are optional and depend on maintainers requesting them. GitLab acts as a CVE Numbering Authority (CNA) and issues CVEs even for minor vulnerabilities. Transparency in security disclosures was considered a positive signal rather than a red flag.

## 5. Defensive Architecture & Isolation

Security must assume compromise is inevitable. Defense-in-depth strategies include:

- Sandboxing untrusted content using iframe sandbox attributes.
- Isolating risky processes with chroot or containerized environments.
- Leveraging microservices to contain blast radius.
- Recognizing that sanitizers are imperfect and subject to parsing discrepancies.

## 6. Cultural & Process Outcomes

The workshop emphasized cultural empowerment. Reviewers must feel confident raising concerns, even when uncertainty is based on intuition. Security responsibility is shared across engineers, security teams, DevOps, and leadership. Dependency additions require deliberate architectural thinking.

***End of Document***