AI-Assisted Secure Code Generation/Remediation

Overview

This capability focuses on leveraging agentic AI to review existing code for vulnerabilities, support pull request (PR) and code review workflows, and analyze dependency and repo health issues. The solution integrates with Git-based repositories (GitHub, GitLab, Bitbucket), CI/CD pipelines, and IDEs to proactively enhance application security.

Objectives

- **Automated Code Review**: Scan existing repositories and new PRs for insecure coding practices and risky patterns.
- **Dependency Analysis**: Identify vulnerable packages, outdated libraries, and license compliance risks.
- **Secure Fix Suggestions**: Generate safe, context-aware code changes and PR comments.
- **CI/CD Workflow Integration**: Seamlessly plug into PR/check-in pipelines for real-time scanning.
- **Repo Health Metrics**: Provide dashboards for technical debt, known vulnerabilities, and unresolved issues.
- **Feedback & Learning Loop**: Incorporate developer review actions to improve model accuracy over time.

Outcomes

- **Rapid AppSec Assessments**: Automate repo-level assessments for client onboarding or periodic health checks.
- **Consulting Efficiency**: Reduce time spent on manual static analysis and code review.
- **Developer Enablement**: Integrate security education and suggestions into developer workflows.
- **Sticky Services**: Offer Git-integrated AI tooling as part of recurring AppSec services.
- **IP & Differentiation**: Build proprietary prompt libraries and AI-assisted workflows for code review and remediation.

Components

Repository Integration Layer

- **Supported Platforms:** GitHub, GitLab, Bitbucket
- **Capabilities:**
    - Scan full repositories or selected branches
    - Comment on PRs with AI-reviewed feedback
    - Auto-generate fix suggestions and create PRs
    - Tag existing issues that reference vulnerable code

Backend AI Engine

- **LLM Interface**: Hosted (OpenAI, Claude) or private (Code Llama, StarCoder) models
- **Prompt Templates**:

    - Review: "Analyze this codebase/file/PR for vulnerabilities and suggest improvements."
    - Dependency: "Identify outdated or insecure dependencies and suggest remediation steps."
    - Fix: "Generate a secure replacement for this code snippet."
- **Static Analysis Tooling**: Integrate Semgrep, Bandit, etc. for pattern-based validation

Dependency & Issue Analyzer

- **SBOM Parsing**: Analyze software bills of materials (CycloneDX, SPDX, Syft, Trivy)
- **Vulnerability Databases**: Integrate with GitHub Advisory DB, OSV, Snyk, or NVD
- **Static Code Analysis:** Integrate with static code analysis tools (CodeQL, Semgrep, Bandit, Brakeman, FindSecBugs, Gitleaks)
- **PR Workflow/CI Integration:** Integration PR workflows (Danger, Reviewdog, Renovate)
- **License Flags**: Detect use of GPL or non-commercial friendly licenses
- **Cross-Referencing**: Link detected issues to existing GitHub/GitLab issue tracker entries

CI/CD & Pull Request Hooks

- Scan PR diffs for code smells, insecure patterns, and dependency risks
- Add inline AI-generated comments with rationale and suggestions
- Optionally auto-block PRs above defined severity thresholds
- Create parallel PRs for automated remediation or dependency updates

Feedback & Learning Loop

- **Developer Actions**: Track accepted/rejected suggestions on PRs
- **Issue Status Monitoring**: Link scanner results to issue resolution timelines
- **Prompt Tuning**: Adapt prompts by language, framework, or repo history

Phased Approach

Phase 1: Prototype

- Build GitHub Action to scan PRs using LLM + Semgrep
- Detect OWASP Top 10 patterns in Python/JavaScript repos
- Leave AI-generated comments on PRs with recommendations

Phase 2: MVP

- Support full repo scanning on demand
- Integrate dependency vulnerability checks (OSV, GitHub Advisory)
- Auto-generate PRs for safe dependency upgrades
- Expand language support (Java, Go, etc.)

Phase 3: Production

- Add GitLab and Bitbucket support
- Customize rule severity thresholds and report formats per client
- Offer remediation-as-code via secure PRs
- Build AppSec dashboard showing repo risk scores and remediation metrics

- Enable client-specific model tuning based on feedback

Implementation Considerations

- **Client data separation:** design for separation of client data, supporting multi-tenancy and/or fully separate environments
- **Security & Privacy**: Do not store or train on proprietary code unless authorized
- **Token Limits**: Chunk large files and use diff-aware prompts for PRs
- **Dependency Depth**: Analyze transitive dependencies where possible
- **Rate Limiting**: Respect Git provider API limits and LLM usage caps
- **Secure Execution**: Run scanners in isolated, non-networked containers to avoid leaking code

Sample Prompts

- **Code Review Prompt**
    - "You are a secure code reviewer. Analyze this pull request diff and comment on any security vulnerabilities or poor coding practices."
- **Dependency Audit Prompt**
    - "Review the following list of dependencies. Identify any that are vulnerable, deprecated, or under restrictive licenses. Suggest replacements."
- **Fix Suggestion Prompt**
    - "You are a secure software engineer. Replace the identified vulnerable function with a secure and efficient alternative. Maintain the same functionality."

Roadmap & Timeline

TBD