

# AI-based code analysis: Safeguarding against credential exposures

Using OpenAI / Ollama Engine



# Table of contents

Abbreviations	3
Introduction	4
Business Challenges	4
Problem statement	5
Solution	5
Design overview	7
Workflow overview	9
Benefits	10
Conclusion	12
References	13
Author information	14

# Abbreviations

Abbreviation	Expansion
LLM	Large Language Model
PR	Pull Request
CI/CD	Continuous Integration/Continuous Deployment
API	Application Programming Interface

# Introduction

In today's fast-paced software development environment, strong security measures are essential. Conventional code security tools frequently do not provide modern enterprises with the flexibility and cost-effectiveness they need. The need for flexible and cost-effective security solutions in development lifecycles has never been higher.

This whitepaper introduces an AI-driven, cost-effective alternative to GitGuardian for code security analysis. Developed as a proof of concept (POC) by Cisco, this solution leverages advanced language models such as Llama3.1:8b and GPT4oMini via the Ollama engine/OpenAI. It is designed to meet the growing needs for customizable security scanning, cost reduction and enhanced productivity in code vulnerability analysis.

Our proposed solution fully complies with the company's security posture, ensuring data localization and eliminating the risk of data exposure to public clouds. Key features include scalable analysis of entire project directories, integration with GPT-based models for in-depth vulnerability detection, customizable configurations for specific security requirements and detailed reporting, enabling developers to identify and address potential security flaws swiftly.

By decreasing dependency on third-party tools such as GitGuardian, this solution not only delivers considerable cost savings but also improves security, productivity and transparency within the code review process. This whitepaper details the innovative architecture of this solution and its financial advantages and demonstrates its efficacy compared to existing solutions.

## Business challenges

In today's competitive business environment, organizations face significant challenges safeguarding their code from vulnerabilities and security threats. These challenges are further complicated by the need to integrate security smoothly into the software development lifecycle without incurring excessive costs. Key business challenges and needs include:

### **1. Cost management:**

Many existing code security solutions, like GitGuardian, involve substantial financial outlays, making it difficult for businesses to maintain cost-effective operations. Organizations need solutions that can provide robust security without straining their budgets.

### **2. Customization and flexibility:**

Businesses require security tools that can be tailored to specific project requirements and integrated into diverse development environments. The inability to customize scanning rules and processes can lead to inefficiencies and overlooked vulnerabilities.

### **3. Early detection of vulnerabilities:**

The need for early detection of security flaws is critical to prevent costly breaches and maintain trust with stakeholders. Implementing effective security measures early in the development process is essential for minimizing risks and ensuring robust protection.

### **4. Seamless integration:**

Security solutions must integrate smoothly with existing workflows and development tools to enhance productivity and reduce manual overhead. Seamless integration helps in maintaining developer efficiency and minimizing disruptions.

### **5. Comprehensive reporting and transparency:**

Detailed and actionable insights are necessary for developers to quickly understand and address security issues. Organizations need transparent reporting mechanisms that provide clear visibility into security vulnerabilities and remediation steps.

Meeting these challenges demands innovative solutions that effectively balance security needs with operational efficiency and cost-effectiveness. By developing a flexible, AI-driven security tool, businesses can better protect their code while optimizing resources and maintaining agility in their development processes.

## Problem statement

In the current technological landscape, the need for a more adaptable and cost-effective solution to detect security vulnerabilities in code has become increasingly critical. With the upcoming requirements of pen-testing increasingly becoming stringent, existing tools may not adequately meet the diverse needs of organizations due to constraints in customization capabilities and pricing structures.

Organizations face the challenge of implementing security measures that are both robust and adaptable to specific project requirements. The limitations of current solutions can lead to inefficiencies and increased operational costs, which undermine the overall security posture and financial sustainability of businesses.

Thus, there is a pressing need for an innovative solution that offers customizable scanning capabilities, reduces dependency on costly third-party tools and integrates seamlessly into existing workflows. This solution must also ensure compliance with security standards while providing detailed feedback and transparency to enable swift identification and remediation of security vulnerabilities.

# Solution

Our solution uses an innovative AI-driven methodology for comprehensive code security analysis, efficiently identifying and reporting vulnerabilities. Here's an overview of the approach:

## 1. Semantic chunking and code analysis:

- The process begins by pulling code directly from repositories and applying semantic chunking to break down files into meaningful segments. This allows for efficient and targeted analysis of code by the Large Language Model (LLM).

## 2. LLM-driven security review:

- The segmented code is then fed into advanced language models, such as those offered by Ollama engine/OpenAI. These models review the code based on preset rules that can be customized to track specific vulnerabilities, including hardcoded passwords, API keys, sensitive tokens and insecure coding practices.
- The LLM provides feedback in a concise format, identifying the problem, the line of code and the line number where the issue is detected.

## 3. Custom rule definitions:

- The solution allows for the extension of its capabilities via Retrieval-Augmented Generation (RAG), enabling the definition of custom rules tailored to specific security requirements. This adaptability ensures that the tool remains relevant to evolving security standards.

## 4. Detailed and customizable reporting:

- Security reports are generated and compiled into a PDF format, categorizing issues by file name and providing line numbers for each identified vulnerability. For complex issues, the solution can also generate potential fixes, significantly reducing the time needed for remediation.
- Reports can be customized with the client's logo, aligning with corporate branding and enhancing the professional presentation of security findings.

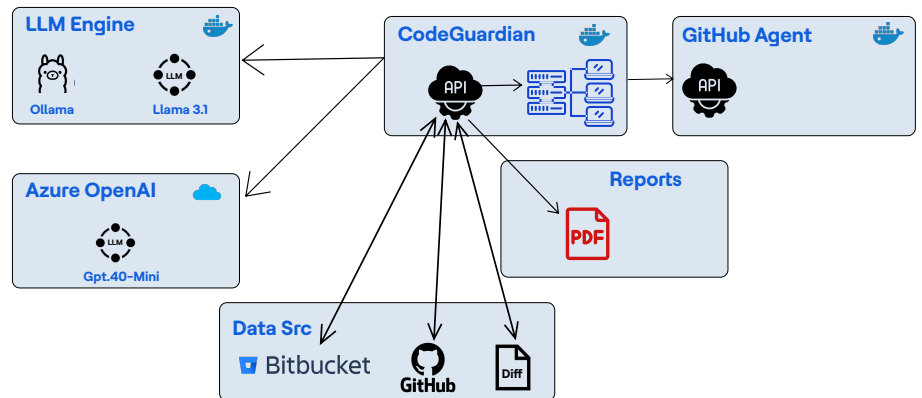
## 5. Integration with development workflows:

- The tool can be augmented to integrate with code review hooks, ensuring that vulnerabilities are identified and reported during the code review process. This proactive approach helps catch security issues at the point of injection, preventing them from propagating into production environments.

## 6. Comprehensive repository-level analysis:

- Operating at the repository level, the solution can identify existing security issues across the entire codebase, providing a thorough assessment of the organization's code security posture. This solution boosts code security through semantic analysis, AI reviews and customizable reporting, enhancing measures in the software development lifecycle.

# Design overview



The proposed system architecture includes several key components:

## 1. FastAPI-based API layer:

The core API is built using FastAPI, which provides fast, asynchronous request handling and allows for easy integration with other tools and services. FastAPI enables the engine to expose multiple endpoints for interacting with the review process, such as submitting scan requests for review or configuring user-specific review preferences.

## 2. Hosting LLM with Ollama in Docker:

- A key aspect of the architecture is the fallback mechanism that ensures the code review engine remains operational even during network outages or cloud API downtimes. The AI engine primarily uses OpenAI's Azure API for code analysis. Still, when connectivity to the cloud is lost or limited, it seamlessly falls back to Ollama, a local language model hosted in Docker containers.
- By leveraging Docker, the system ensures that the local instance of Ollama is isolated and resource-efficient. This Docker-based deployment of Ollama serves as a fallback LLM for local inferencing, enabling AI-driven code reviews even when cloud access is unavailable.

## 3. Bitbucket integration and callback hooks:

The system integrates with Bitbucket repositories using webhooks, which allow the engine to automatically initiate scans as part of the CI/CD lifecycle. When a pull request is created or updated in Bitbucket, the system triggers a webhook that sends the code diff to the code guardian container for analysis. Feedback is provided through a callback hook setup, updating Bitbucket with comments, inline suggestions and review status.

## 4. Configurable templates and customization:

The engine's scan criteria can be customized using templates that adapt to specific coding standards, security policies, or project guidelines. This allows teams to configure the engine based on their needs, tailoring it to different types of projects, languages, or compliance requirements

## Pluggable LLM: Enhancing the code security engine

The integration of a pluggable LLM into the code security engine via OpenAI configuration or by deploying the necessary model to the Ollama container introduces significant advancements. This enhancement supports the identification of security vulnerabilities and aligns with the diverse needs of development teams. Here's how this feature substantially contributes to the code security process:

### 1. Consistency in security assessments:

The primary benefit of employing a pluggable LLM is its capacity to deliver consistent quality in security assessments. Unlike human reviewers, who may have varying expertise and focus, an LLM provides a standardized approach to evaluating code security. This ensures that every piece of code is scrutinized against uniform criteria, leading to consistent detection of vulnerabilities and adherence to security best practices. By minimizing reliance on individual reviewers, teams can avoid inconsistencies often encountered in manual security assessments.

### 2. Adaptability to organizational needs:

The pluggable nature of the LLM allows it to adapt to the evolving security requirements of an organization. As new vulnerabilities, coding standards, or compliance regulations emerge, the LLM can be updated or swapped out without major modifications to the underlying architecture. This adaptability enables organizations to stay current with security advancements and continuously integrate the latest practices in vulnerability management.

### 3. Polyglot security analysis:

Modern LLMs are inherently polyglot, allowing them to understand and analyze code written in multiple programming languages. This versatility enables a single LLM to handle various technology stacks, including:

- **Infrastructure as code:** Reviewing configurations and scripts in Terraform or Docker Compose for security risks.
- **Backend development:** Analyzing codebases in Python and Java, providing context-sensitive security feedback tailored to each language's paradigms.
- **Frontend frameworks:** Evaluating code written in React, Angular and other frontend technologies to ensure UI components adhere to security best practices.

This capability streamlines the security process and eliminates the need for multiple engines tailored to specific languages or frameworks, enhancing efficiency.

### 4. Reducing workload on security teams:

Manual security reviews can be time-consuming and often place significant pressure on security personnel. By implementing an LLM within the code security engine, routine security tasks are automated, allowing security experts to focus on more critical activities, such as threat modeling or incident response. This workload distribution not only boosts productivity but also helps prevent burnout among team members, fostering a healthier work environment.

## 5. Scalable solution for expanding teams:

As development teams grow and the volume of code increases, maintaining a high standard of security assessment becomes challenging. The asynchronous processing capabilities of the LLM enable it to handle large volumes of code simultaneously, ensuring that security feedback is timely and relevant. This scalability makes the code security engine an ideal solution for organizations of all sizes, ranging from startups to large enterprises.

# Workflow overview

The AI-driven code security analysis workflow is designed to seamlessly integrate with the software development lifecycle, providing automated and comprehensive security reviews of code repositories. Utilizing advanced language models and semantic chunking, the workflow ensures the identification and remediation of security vulnerabilities, thereby enhancing the overall security posture of the organization.

## 1. Code extraction and preparation

- **Repository access:** The system accesses the source code repository to extract the latest codebase for analysis.
- **Semantic chunking:** The extracted code is semantically chunked, dividing it into manageable and meaningful segments to facilitate focused analysis.

## 2. Security analysis initiation

- **LLM invocation:** The system invokes AI models, such as those available through the Ollama engine or Azure OpenAI, to begin security analysis on the prepared code chunks.
- **Custom rule application:** Predefined security rules, which can be customized via RAG, are applied to each segment, setting the framework for vulnerability detection.

## 3. Vulnerability detection and feedback generation

- **Detailed analysis:** Each code chunk is analyzed for potential security issues, including hardcoded credentials, API keys, sensitive tokens and insecure practices.
- **Feedback compilation:** The AI model generates concise feedback for detected vulnerabilities, providing a description of the issue and specifying the line number and code segment involved.

## 4. Report generation

- **Comprehensive reporting:** The results from the analysis are compiled into a detailed report, categorizing issues by file name and line number. In complex cases, the system suggests potential fixes to expedite resolution.
- **Custom branding:** The report can be customized with the client's logo, ensuring alignment with corporate branding standards.

## 5. Integration with development workflows

- **Code review augmentation:** The generated reports are integrated into the existing code review process and attached directly to code reviews to facilitate timely identification and remediation of vulnerabilities.
- **Repository-level analysis:** The tool operates at the repository level, ensuring that both new and existing issues are thoroughly identified and addressed.

## 6. Iterative feedback and resolution

- **Developer interaction:** Developers receive the security analysis report within their code review interface and can address the identified vulnerabilities by making necessary code changes.
- **Continuous analysis:** As code changes are made and new commits are pushed, the system automatically re-initiates analysis to ensure ongoing security compliance.

## 7. Finalization and continuous improvement

- **Security clearance:** Once all identified issues are resolved, the code is cleared for further stages in the development lifecycle.
- **Metrics collection and reporting:** The system collects data on the analysis process, including time taken for reviews, common vulnerabilities and trends in security improvements. Regular reports are generated for stakeholders to ensure continuous security enhancement efforts.

This workflow not only strengthens the security of codebases but also enhances developer productivity by automating the detection and remediation of vulnerabilities, ensuring that security is maintained throughout the development process.

# Benefits

The AI-driven code security solution offers numerous advantages that address both security and operational needs within an organization. Here are the key benefits:

### 1. Cost-effectiveness:

By reducing reliance on third-party tools like GitGuardian, the solution presents significant cost savings. It minimizes operational expenses without sacrificing the quality and comprehensiveness of security analysis.

### 2. Enhanced security:

The tool effectively detects a wide range of security vulnerabilities, including hardcoded passwords, API keys and insecure coding practices. This proactive identification helps in mitigating potential security risks before they escalate into serious breaches.

### **3. Customization and flexibility:**

With customizable scanning rules and the ability to define custom security prompts, the solution is highly adaptable to specific organizational requirements. This flexibility ensures that the tool remains relevant and effective across different projects and security landscapes.

### **4. Scalable and comprehensive analysis:**

Capable of processing entire project directories and analyzing commonly used file types, the solution provides a thorough assessment of code security across the entire codebase. This scalability is essential for large-scale projects and complex code environments.

### **5. Detailed reporting and transparency:**

The generation of detailed per-file security reports, combined into comprehensive PDF documents, provides clear and actionable insights. These reports facilitate quick identification and remediation of vulnerabilities, enhancing the overall transparency of the security review process.

### **6. Seamless integration:**

The solution integrates effortlessly with existing development workflows, including code review processes. Generating reports and attaching them to code reviews ensures that security issues are caught and addressed at the point of injection.

### **7. Increased productivity:**

By automating the vulnerability detection process, the tool reduces the need for manual code reviews, freeing up developer time and increasing productivity. The generation of potential fixes for complex issues further accelerates the remediation process.

### **8. Deployment flexibility:**

With the ability to run in both cloud (e.g., Azure OpenAI) and on-premises environments, the solution offers flexibility in deployment, allowing organizations to choose infrastructure setups that align with their security and compliance needs.

### **9. Corporate branding customization:**

The ability to customize reports with the client's logo allows the solution to fit seamlessly into the organization's branding scheme, providing a professional and cohesive presentation of security findings.

Overall, this solution enhances the security and efficiency of the software development process, empowering organizations to maintain a robust security posture while optimizing resource allocation and operational workflows.

# Conclusion

In an era where software security is paramount, organizations must adopt innovative solutions that align with their dynamic needs and budget constraints. Our AI-driven code security solution emerges as a transformative tool that not only addresses these challenges but also enhances the overall development lifecycle. By leveraging advanced language models and semantic analysis, the solution provides a comprehensive, scalable and flexible approach to detecting and mitigating code vulnerabilities.

This solution reduces operational costs by minimizing dependence on third-party tools like GitGuardian while offering customizable security checks that can adapt to specific organizational requirements. Its seamless integration into existing workflows ensures that security issues are identified and addressed promptly, thereby maintaining the integrity of the codebase.

Moreover, the deployment flexibility across cloud and on-prem environments, along with the capacity for corporate branding, ensures that the solution meets diverse infrastructure and branding needs. With detailed reporting and the ability to suggest complex fixes, the tool not only enhances security but also boosts productivity by streamlining the remediation process.

In conclusion, this AI-driven approach empowers organizations to secure their software development environments efficiently and effectively, safeguarding their digital assets while optimizing resources. As the landscape of cybersecurity continues to evolve, our solution positions businesses at the forefront of proactive security management, ready to meet the challenges of tomorrow with confidence and agility.

In conclusion, as the complexity of software systems increases and the demand for rapid innovation grows, organizations must embrace AI-driven solutions that augment human capabilities rather than replace them. An AI code review assistant represents a technological advancement and a strategic necessity for modern software development. By investing in such an intelligent solution, teams can not only enhance their coding standards and quality but also cultivate an agile, collaborative culture that positions them for success in an increasingly competitive landscape. The future of coding is not merely about writing code; it's about leveraging intelligence to unlock new levels of creativity, efficiency and excellence in software development.

# References

- <https://ollama.com/blog/ollama-is-now-available-as-an-official-docker-image>
- <https://learn.microsoft.com/en-us/azure/ai-services/openai/overview>
- <https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/create-resource?pivots=web-portal>
- <https://dev.to/willvelida/building-nlp-applications-with-azure-openai-2637>
- <https://github.com/Azure-Samples/openai>
- <https://www.gitguardian.com/monitor-internal-repositories-for-secrets>

# Author information



## **Karthik Subramanian**

Karthik Subramanian has 19 years of experience building enterprise applications, focusing on network management solutions. His expertise includes developing reliable and scalable applications using Java technologies, AWS, Networking, AI and Python. Karthik is dedicated to creating practical, efficient systems that support network operations and management.

# HCLTech | Supercharging Progress™

HCLTech is a global technology company, home to more than 223,000 people across 60 countries, delivering industry-leading capabilities centered around digital, engineering, cloud and AI, powered by a broad portfolio of technology services and products. We work with clients across all major verticals, providing industry solutions for Financial Services, Manufacturing, Life Sciences and Healthcare, Technology and Services, Telecom and Media, Retail and CPG and Public Services. Consolidated revenues as of 12 months ending March 2025 totaled \$13.8 billion. To learn how we can supercharge progress for you, visit [hcltech.com](https://hcltech.com).

[hcltech.com](https://hcltech.com)

