# Static Analysis & Secrets Scanning – Detailed Guide with Real-Time Example

## 🧪 What is Static Analysis?

**Static Analysis** is the process of examining source code **without executing the program** to detect errors, vulnerabilities, or non-compliance with coding standards.

It is commonly used in early development stages to improve code quality and enforce security standards.

## 🔍 Key Features of Static Analysis

* Detects syntax errors, logic bugs, and potential vulnerabilities
* Enforces coding standards (e.g., PEP8, MISRA)
* Integrates into CI/CD pipelines
* Tools: **SonarQube, Veracode, Fortify SCA, Checkmarx**

### Real-Time Example:

A development team integrates **SonarQube** into Jenkins. On every code commit: 1. SonarQube performs a scan. 2. Flags a hard-coded SQL query without input validation. 3. Highlights a potential **SQL Injection** issue. 4. Developer refactors the code with prepared statements.

## 🔐 What is Secrets Scanning?

**Secrets Scanning** detects accidentally exposed **credentials or sensitive information** in source code repositories or configuration files.

These secrets can include: - API keys - Passwords - Access tokens - Private keys

## ⚠️ Risks of Leaking Secrets

* Unauthorized access to infrastructure (e.g., AWS, GCP)
* Exposure of confidential user data
* Service disruptions or abuse
* Compliance violations (e.g., GDPR, HIPAA)

## 🛠️ Popular Secrets Scanning Tools

| Tool | Description |
| --- | --- |
| **GitGuardian** | Scans Git repositories for API keys and credentials |
| **TruffleHog** | Searches commit history for high-entropy secrets |
| **Gitleaks** | Fast, customizable, open-source secrets scanner |
| **Talisman** | Pre-commit hook to stop secrets before committing |

## 🔁 Real-Time Example: API Key Exposure

1. A developer accidentally commits a GitHub access token to a public repo.
2. GitGuardian scans the repo and sends an alert.
3. Token is rotated, and the repo history is rewritten to remove the secret.
4. A pre-commit hook using **Talisman** is added to prevent future leaks.

## 🔄 Integration into DevSecOps Pipeline

| Phase | Static Analysis | Secrets Scanning |
| --- | --- | --- |
| Development | IDE plugins (e.g., SonarLint) | Pre-commit hooks (Talisman, Gitleaks) |
| CI/CD | SAST via Jenkins, GitLab CI | Secrets scanning tools integrated in CI |
| Code Review | Code quality and security checks | Manual/automated review for secrets |
| Post-Deployment | Periodic scanning of repos | Alerts on newly added secrets |

## ✅ Best Practices

* Use **automated SAST tools** during development
* Enforce **code reviews** with security focus
* Add **pre-commit hooks** to detect secrets before commit
* Rotate credentials frequently
* Never hardcode secrets — use vaults or secret managers

## 🧩 Conclusion

Combining **Static Analysis** and **Secrets Scanning** enables early detection of vulnerabilities and prevents serious security incidents. These practices are essential for modern secure software development and should be integrated across the SDLC.

Would you like this document exported as a PDF or Word file?