

Static Application Security Testing SAST Practice Training Course

17 – 21 May 2027
Jakarta

AGILE LEADERS
Training Center

AGILE LEADERS
Training Center

Static Application Security Testing SAST Practice Training Course

Ref: 2150_257483 **Date:** 17 – 21 May 2027 **Location:** Jakarta **Fees:** 5700 Euro

Overview

Static Application Security Testing SAST Practice Training Course is a one-week foundation course for developers, application security analysts, DevSecOps engineers, quality teams, and technical leads, who leave with a SAST Adoption, Triage and Remediation Workflow. Participants practice static code analysis, SAST pipeline integration, security finding triage, false-positive management, and remediation tracking across software delivery. Agile Leaders Training Center presents Static Application Security Testing practice.

Who Should Attend

- Developers responsible for building and maintaining application code
- Security personnel responsible for application findings and secure development support
- DevSecOps teams responsible for build and delivery pipelines
- Quality teams responsible for test evidence and release readiness
- Technical leaders responsible for SAST policy, adoption, and improvement

The course assumes participants can read basic source code and development workflows, and it leaves penetration testing, exploit development, tool administration at scale, and language-specific secure coding mastery outside its scope.

Departments and Industries

The course supports departments and sectors that develop, acquire, test, or maintain software.

- Software engineering and application development
- Application security and product security
- DevSecOps, platform engineering, and release management
- Quality assurance and software testing
- Technology vendors and internal digital-product teams

Learning Objectives

By the end of this course, participants will be able to:

- Explain SAST methods, strengths, limitations, and complementary tests.
- Configure a scoped scan and establish an initial baseline.
- Integrate SAST feedback into developer and CI/CD workflows.
- Review findings using data flow, context, severity, and exploitability.
- Govern false positives, suppressions, remediation, and exceptions.

- Build a SAST Adoption, Triage and Remediation Workflow.

Course Agenda

Day 1: Understand Static Security Analysis

- SAST Position in the Secure Development Lifecycle Map
- Source Code, Intermediate Representation, and Rule Analysis Model
- Pattern, Data-Flow, and Taint-Analysis Concept Map
- SAST Strength, Limitation, False-Positive, and False-Negative Matrix
- SAST, DAST, SCA, and Manual Review Comparison Framework

Day 2: Scope and Configure Scanning

- Application, Repository, Language, and Framework Inventory
- Tool Capability and Coverage Evaluation Checklist
- Scan Scope, Branch, Build, and Dependency Configuration
- Rule Set, Severity, and Policy Baseline
- Initial Scan Quality and Coverage Review

Day 3: Integrate SAST into Delivery

- IDE, Commit, Pull-Request, and Pipeline Feedback Design
- CI/CD Security Gate and Failure-Condition Matrix
- Scan Scheduling and Incremental Analysis Workflow
- Developer Notification and Finding Ownership Model
- Pipeline Performance and Exception Handling Plan

Day 4: Triage and Remediate Findings

- Source-to-Sink and Data-Flow Finding Review
- Code Context, Reachability, and Exploitability Assessment
- False-Positive Validation and Suppression Governance
- Remediation Priority and Service-Level Decision Matrix
- Developer-Security Collaboration and Closure Evidence

Day 5: Practice SAST Operations

- Suggested Exercise: Scope a Static Analysis Scan
- Suggested Exercise: Triage a Data-Flow Finding
- Suggested Exercise: Review a False-Positive Claim
- Suggested Exercise: Design a CI/CD Security Gate
- Capstone Exercise: Complete a SAST Adoption, Triage and Remediation Workflow

AGILE LEADERS
Training Center

Practical Exercises

The course uses suggested activities based on application repositories, scan results, pipeline decisions, and remediation records.

- Suggested activity: map application languages, frameworks, repositories, build requirements, and SAST coverage limits.
- Suggested activity: trace an untrusted data path and assess the code context, reachability, severity, and remediation evidence.
- Suggested activity: distinguish an actionable issue from a false positive and document any approved suppression.
- Suggested activity: design ownership, gates, exceptions, measures, and continuous-improvement actions for SAST operations.

FAQs

Who suits Static Application Security Testing practice training, and what does it assume?

Static Application Security Testing practice training suits developers, application security analysts, DevSecOps engineers, quality teams, and technical leads. It assumes basic source-code and development-workflow literacy.

How does SAST differ from dynamic application security testing?

SAST examines source code or compiled representations without running the application. Dynamic testing evaluates a running application and can observe runtime behavior that static analysis may not represent.

Why do SAST findings require manual triage?

SAST findings require triage because a tool identifies potential issues from rules and code models, but application context, reachability, safeguards, business logic, and exploitability determine whether a finding is actionable.

Should every SAST finding block a CI/CD pipeline?

Not every SAST finding should block delivery. Teams should define gates using validated severity, confidence, policy, application risk, baseline status, exploitability context, and an authorized exception process.

How should SAST false positives be governed?

False positives should be validated, documented, assigned a reason and owner, reviewed before suppression, linked to the affected rule and scope, and periodically reconsidered when code, tools, rules, or risk changes.

AGILE LEADERS
Training Center

Conclusion

Participants take back a SAST Adoption, Triage and Remediation Workflow. It combines scope, configuration, pipeline integration, findings review, false-positive governance, remediation, exceptions, evidence, and metrics. The workflow changes disconnected scan output into an accountable development process that gives teams timely feedback while preserving contextual review and complementary security testing.