Logotipo, nome da empresa

Descrição gerada automaticamente

**ANHANGUERA EDUCACIONAL**

**UNIDADE SANTANA**

**Curso de Tecnologia em Análise e Desenvolvimento de Sistemas**

**Thiago Melo da Silva – RA: 401966816563**

**ANÁLISE ESTÁTICA DE CÓDIGO: COMPARAÇÃO ENTRE AVALIAÇÃO MANUAL E AUTOMATIZADA**

**São Paulo**

**2024**

Sumário

[**Introdução** 7](#_Toc198814195)

[**Desenvolvimento** 8](#_Toc198814196)

[**Conclusão** 10](#_Toc198814197)

[**Referências** 11](#_Toc198814198)

# **Introdução**

A segurança no desenvolvimento de software é uma preocupação crescente em um cenário onde falhas podem comprometer dados sensíveis, causar prejuízos financeiros e afetar a imagem de empresas. Entre as estratégias para assegurar sistemas mais seguros, destaca-se a análise estática de código, também conhecida como Static Application Security Testing (SAST). Essa abordagem consiste na inspeção do código-fonte do software sem sua execução, identificando vulnerabilidades desde as fases iniciais do desenvolvimento. A análise pode ser realizada de forma manual, por revisores humanos, ou por ferramentas automatizadas. Este trabalho tem como objetivo discutir as vantagens e desvantagens dessas duas abordagens, evidenciando sua importância no ciclo de desenvolvimento seguro.

# **Desenvolvimento**

A análise estática de código, ou SAST (Static Application Security Testing), é uma técnica de segurança que visa identificar vulnerabilidades no código-fonte de uma aplicação antes de sua execução. Essa análise pode ser realizada de forma manual ou automatizada, e ambas as abordagens apresentam vantagens e desvantagens que impactam diretamente a qualidade, o tempo de desenvolvimento e os custos de manutenção de sistemas.

A análise manual é realizada por profissionais de desenvolvimento ou especialistas em segurança da informação, que inspecionam o código em busca de falhas de lógica, má utilização de bibliotecas, práticas inseguras de programação ou violações de padrões da empresa. Esse tipo de análise permite uma compreensão mais profunda do contexto do sistema e das intenções do programador, o que favorece a identificação de vulnerabilidades complexas, como problemas de lógica de negócio, autenticação mal implementada ou falhas específicas que fogem aos padrões conhecidos. Além disso, durante uma revisão manual, é possível que os revisores sugiram melhorias estruturais e práticas mais seguras, contribuindo para o amadurecimento técnico da equipe.

Entretanto, esse processo é extremamente dependente da experiência dos profissionais envolvidos. Revisões manuais podem ser demoradas, especialmente em sistemas com grande volume de código ou com histórico de baixa padronização. Além disso, fatores humanos como fadiga, distrações ou excesso de confiança podem gerar inconsistências na análise, fazendo com que falhas passem despercebidas.

Já a análise estática automatizada utiliza ferramentas que escaneiam o código-fonte em busca de vulnerabilidades conhecidas, padrões inseguros e violações de boas práticas. Ferramentas como SonarQube, Checkmarx, Fortify e Brakeman são amplamente utilizadas no mercado para realizar essa tarefa com agilidade e precisão. A principal vantagem dessa abordagem é a velocidade: grandes volumes de código podem ser verificados rapidamente, inclusive de forma contínua, integrando-se aos pipelines de integração e entrega contínuas (CI/CD). Isso permite que falhas sejam detectadas logo após o commit do desenvolvedor, reduzindo o custo de correção e evitando que vulnerabilidades cheguem a ambientes produtivos.

Além da velocidade, a padronização das análises e a geração automática de relatórios ajudam na documentação do processo de segurança e facilitam auditorias. Porém, as ferramentas automatizadas tendem a apresentar falsos positivos — indicando como vulneráveis trechos de código que não representam riscos reais — ou a não detectar falhas mais sofisticadas, que requerem interpretação de contexto. Também é importante considerar que essas ferramentas precisam ser configuradas corretamente e atualizadas regularmente para manter sua eficácia.

Por esse motivo, o uso isolado de uma das abordagens pode não ser suficiente. A análise manual é mais eficaz para aspectos qualitativos e contextuais, enquanto a análise automatizada cobre melhor as falhas padronizadas e recorrentes. O uso combinado — também conhecido como abordagem híbrida — é o mais recomendado, pois une os pontos fortes de cada método. Por exemplo, um projeto pode utilizar ferramentas automatizadas em cada push de código e, periodicamente, realizar revisões manuais nas partes mais críticas da aplicação, como autenticação, controle de acesso e tratamento de dados sensíveis.

Além disso, a análise estática não deve ser a única técnica de verificação de segurança. Ela deve ser complementada por testes dinâmicos (DAST), análise de dependências, revisões de arquitetura e testes de penetração (pentests), compondo um ciclo de desenvolvimento seguro (Secure Software Development Lifecycle – SSDLC).

# **Conclusão**

A análise estática de código é uma prática essencial para garantir a segurança de aplicações, sendo útil tanto em fases iniciais quanto após a implantação. A avaliação manual oferece maior profundidade e contextualização, enquanto a automatizada assegura rapidez e constância nos testes. Considerando os prós e contras de cada método, a combinação estratégica dos dois permite fortalecer a segurança do software de maneira eficiente e econômica. Dessa forma, equipes de desenvolvimento podem mitigar vulnerabilidades de maneira proativa, elevando a qualidade e a confiabilidade de seus sistemas.

# **Referências**

BRASIL. Certifiquei. SAST: o que é análise estática e como aplicá-la em segurança de aplicações. Disponível em: https://certifiquei.com.br/sast-analise-estatica/. Acesso em: 22 maio 2025.

TEXTO SEGURO. Diferenças entre análise estática e dinâmica de código. Disponível em: https://textoseguro.com.br/analise-estatica-vs-analise-dinamica/. Acesso em: 22 maio 2025.

ALURA. O que é SAST (Static Application Security Testing)? Disponível em: https://www.alura.com.br/artigos/sast-static-application-security-testing. Acesso em: 22 maio 2025.