ESCOLA SUPERIOR DO PARLAMENTO CEARENSE - UNIPACE

USO DE FERRAMENTAS SAST PARA DETECÇÃO PREVENTIVA DE VULNERABILIDADES

Valdeclebio Farrapo Costa

Felipe Lira Formiga Andrade

RESUMO

A crescente digitalização da gestão pública, acompanhada pelo uso intensivo de sistemas e aplicativos, tem ampliado a superfície de ataque e elevado o risco de incidentes de segurança. Nesse cenário, ferramentas de análise estática de segurança de aplicações (Static Application Security Testing – SAST) surgem como alternativas eficientes para identificar vulnerabilidades ainda na fase de desenvolvimento, reduzindo custos e prevenindo falhas em produção. Este trabalho apresenta uma avaliação comparativa entre duas ferramentas SAST open-source e standalone — Flawfinder e Visual Code Grepper (VCG) — na detecção de vulnerabilidades em projetos C/C++, com foco na classificação Common Weakness Enumeration (CWE), mantida pelo MITRE Corporation. Foram analisados 30 repositórios públicos do GitHub, totalizando 120 snapshots de código, considerando métricas de volume total de vulnerabilidades, distribuição por severidade e desempenho por categoria CWE. Os resultados indicaram que o Flawfinder apresentou maior eficácia na detecção de falhas relacionadas à manipulação de memória (como CWE-119 e CWE-120), enquanto o VCG destacou-se em vulnerabilidades ligadas à manipulação de arquivos e recursos (como CWE-404). A pesquisa confirma que o uso combinado dessas ferramentas amplia a cobertura de detecção, sendo recomendada para ambientes críticos e de interesse público.

Palavras-chave: SAST; Segurança de Software; CWE; Flawfinder; Visual Code Grepper; Vulnerabilidades em C/C++.

1 INTRODUÇÃO

A transformação digital na gestão pública tem se intensificado nos últimos anos, com a adoção crescente de sistemas eletrônicos, aplicativos e plataformas online para a oferta de serviços essenciais à população, como saúde, educação, segurança pública, assistência social e transparência governamental. Essa modernização tecnológica, embora traga ganhos significativos de eficiência, acessibilidade e economicidade, também amplia exponencialmente a superfície de

ataque dos sistemas. O aumento de invasões, fraudes, vazamentos de dados e golpes virtuais tem colocado em evidência a importância da segurança da informação como pilar essencial para a continuidade e confiabilidade dos serviços prestados.

A segurança de software começa na sua concepção e se estende por todo o ciclo de desenvolvimento. Nesse cenário, padrões e bases de conhecimento são indispensáveis para orientar a detecção e a mitigação de vulnerabilidades. Um exemplo fundamental é a CWE (Common Weakness Enumeration), um catálogo padronizado de fraquezas de software mantido e administrado pelo MITRE Corporation, uma organização sem fins lucrativos que apoia a segurança cibernética global. A CWE não lista ataques ou falhas específicas, mas sim categorias de fraquezas que podem ser exploradas, como “Buffer Overflow” ou “SQL Injection”. Essa padronização é amplamente utilizada por desenvolvedores, analistas de segurança e fabricantes de ferramentas para criar relatórios consistentes e comparáveis, além de embasar práticas de teste, auditoria e treinamento.

Em paralelo à CWE, há também a CVE (Common Vulnerabilities and Exposures), igualmente administrada pelo MITRE, mas com um foco distinto. Enquanto a CWE descreve tipos genéricos de fraqueza, a CVE lista vulnerabilidades específicas e já identificadas em softwares ou dispositivos, atribuindo a cada uma um identificador único. Por exemplo, “CWE-120” refere-se a uma fraqueza genérica de “Stack-Based Buffer Overflow”, ao passo que “CVE2024-12345” representaria uma ocorrência real e específica desse tipo de falha em determinado sistema. Em resumo: CWE classifica fraquezas conceituais; CVE cataloga vulnerabilidades concretas encontradas no mundo real.

Outro ator central no ecossistema de segurança é a OWASP (Open Web Application Security Project), uma comunidade global sem fins lucrativos dedicada a melhorar a segurança de aplicações. A OWASP mantém diretrizes, ferramentas e listas de referência amplamente adotadas, como o OWASP Top 10, que destaca as vulnerabilidades mais críticas em aplicações web. Essas referências orientam

equipes de desenvolvimento e segurança na priorização de riscos e na adoção de práticas defensivas.

No campo das técnicas de detecção de vulnerabilidades, destacam-se três abordagens principais: a análise estática, a análise dinâmica e a análise híbrida. A literatura de engenharia de software e segurança, especialmente em manuais consolidados como o OWASP Testing Guide e relatórios do NIST sobre avaliação de segurança de software, destaca consistentemente as diferenças entre esses métodos, evidenciando que cada abordagem oferece níveis distintos de profundidade, cobertura e precisão na identificação de falhas. Essa distinção fundamenta a importância de comparar ferramentas SAST dentro de um escopo metodológico claro:

* SAST (Static Application Security Testing): Análise estática do código-fonte,bytecode ou binário, sem execução do programa. Permite identificar vulnerabilidades ainda na fase de desenvolvimento, reduzindo o custo de correção e evitando que falhas cheguem ao ambiente de produção.

* DAST (Dynamic Application Security Testing): Análise dinâmica realizada durantea execução da aplicação, simulando ataques externos para identificar falhas exploráveis em tempo real, sem acesso ao código-fonte.

* IAST (Interactive Application Security Testing): Combina elementos de SAST eDAST, realizando análises dinâmicas com instrumentação interna da aplicação em execução, permitindo identificar vulnerabilidades com maior precisão e contexto.

Essas abordagens são complementares: enquanto o SAST atua na prevenção, o DAST detecta problemas que se manifestam somente em execução, e o IAST oferece visibilidade profunda sobre vulnerabilidades efetivamente exploráveis.

Entre as ferramentas de SAST open-source voltadas para código C/C++, destacamse:

* Flawfinder: ferramenta de linha de comando que analisa código-fonte em C e C++procurando padrões perigosos que possam resultar em vulnerabilidades. Baseia-se em uma lista de funções conhecidas por apresentarem riscos e associa cada achado a um nível de severidade e a um identificador CWE correspondente. Sua execução é rápida, simples e não requer compilação do código.

* Visual Code Grepper (VCG): ferramenta com interface gráfica que realizavarreduras no código-fonte em busca de padrões vulneráveis. Além de C e C++, também suporta outras linguagens. Permite personalização de regras e, embora não atribua automaticamente CWEs, possibilita mapeamento manual dessas fraquezas para padronização dos resultados.

Considerando a importância dessas ferramentas e padrões, este estudo investiga a eficácia do Flawfinder e do VCG na detecção de vulnerabilidades categorizadas pela CWE em projetos C/C++ hospedados no GitHub. O objetivo é fornecer dados comparativos que auxiliem gestores de TI, desenvolvedores e especialistas em segurança — especialmente no setor público — a selecionar estratégias e ferramentas adequadas para a proteção de sistemas críticos. Essa investigação se torna particularmente relevante diante da escassez de estudos que avaliem, de forma sistemática, ferramentas SAST open-source aplicadas especificamente a código C/C++ utilizado em ambientes governamentais, além da pouca disponibilidade de análises comparativas que correlacionem detecções com categorias CWE de maneira estruturada. Assim, o estudo contribui para preencher essas lacunas e apoiar decisões técnicas fundamentadas.

2 FUNDAMENTAÇÃO TEÓRICA E/OU DESENVOLVIMENTO

A segurança da informação tem se consolidado como uma das áreas mais sensíveis e estratégicas da gestão pública no contexto digital contemporâneo. Com a crescente digitalização de serviços, o Estado brasileiro e diversas instituições governamentais passaram a depender cada vez mais de sistemas computacionais e aplicações web para a execução de atividades essenciais. Essa transição, embora represente um avanço na eficiência dos serviços públicos, também ampliou a exposição dessas estruturas a ataques cibernéticos.

Autores como Ahmed (2019) e Dias et al. (2023) destacam que a falta de investimentos em mecanismos preventivos de segurança resulta em danos financeiros, institucionais e sociais, especialmente quando ocorrem vazamentos de dados pessoais, interrupções de serviços ou manipulação indevida de informações governamentais. Nesse sentido, é essencial que o ciclo de desenvolvimento de software adote práticas seguras desde as etapas iniciais da codificação. Uma das abordagens mais eficazes é o uso de ferramentas SAST (Static Application Security Testing), que analisam o código-fonte de forma estática, ou seja, sem executá-lo, identificando vulnerabilidades potenciais com agilidade e em estágios ainda corrigíveis.

Segundo o OWASP (Open Web Application Security Project), as ferramentas SAST devem ser integradas à rotina de desenvolvimento, oferecendo relatórios que possibilitem o tratamento rápido de falhas de segurança. Muitas ferramentas do mercado, entretanto, são comerciais e requerem o envio do código-fonte para servidores externos, o que representa um obstáculo para aplicações críticas, especialmente no setor público, onde a confidencialidade do código e dos dados é um requisito inegociável.

Neste contexto, ferramentas open-source e standalone, que funcionam localmente sem a necessidade de upload do código para servidores externos, representam uma alternativa segura e acessível. Dentre elas, destacam-se o Flawfinder e o Visual Code Grepper (VCG), que são compatíveis com C/C++ e

foram projetadas para identificar padrões de código potencialmente vulneráveis com base em catálogos como o Common Weakness Enumeration (CWE).

No caso do Flawfinder, os relatórios já trazem automaticamente os CWEs associados a cada falha identificada. Já o VCG exige análise manual para correlacionar cada item detectado com as CWEs correspondentes. Essa etapa foi fundamental para permitir uma comparação precisa e consistente entre as ferramentas.

Ao adotar essa metodologia, foi possível analisar não apenas o número total de vulnerabilidades identificadas, mas também a especialização de cada ferramenta em tipos específicos de falhas, o que representa um diferencial em relação a estudos anteriores. Os dados obtidos fornecem subsídios importantes para que gestores públicos e profissionais de segurança possam escolher ferramentas mais adequadas à realidade das aplicações que mantêm, aumentando a resiliência dos sistemas públicos frente à crescente ameaça dos crimes cibernéticos.

3 METODOLOGIA

Este estudo adota uma abordagem empírica e exploratória com foco na comparação da eficácia de ferramentas SAST open-source e standalone na detecção de vulnerabilidades em projetos desenvolvidos em C/C++. A metodologia foi dividida em três etapas principais: seleção de ferramentas, escolha dos projetos e análise dos resultados com base na classificação CWE.

3.1 Seleção das Ferramentas

Foram definidos critérios rigorosos para a escolha das ferramentas SAST utilizadas na análise:

- Código aberto (open-source): para garantir acesso ao funcionamento interno da ferramenta.

- Execução local (standalone): para evitar riscos relacionados ao envio de código-fonte para servidores externos.
- Compatibilidade com C/C++: linguagens comumente usadas em sistemas críticos, especialmente no setor público.
- Documentação ativa e acessível: para viabilizar o uso prático e a replicação dos testes.

Com base nesses critérios, as ferramentas selecionadas foram:

- Flawfinder
- Visual Code Grepper (VCG)

3.2 Seleção dos Repositórios

A coleta de dados foi realizada com base em projetos disponíveis publicamente no GitHub. Os critérios de seleção foram:

- Código majoritariamente em C ou C++ (≥ 70%);
- Acesso aberto ao código-fonte e ao histórico de commits;
- Variedade de contextos e tamanhos dos projetos, para ampliar a representatividade da análise.

No total, 30 repositórios foram escolhidos. Para cada um, foram analisadas quatro versões distintas do código (snapshots): o primeiro commit com código útil, os commits do primeiro e terceiro quartis do histórico do projeto, e o commit mais recente. Isso resultou em 120 snapshots de código.

3.3 Execução das Ferramentas

Cada snapshot foi analisado separadamente pelas duas ferramentas. A execução seguiu as seguintes etapas:

- Flawfinder: executado via linha de comando, com relatório em formato de tabela, incluindo severidade e CWEs.

- VCG: executado por interface gráfica, com exportação dos resultados. Como a ferramenta não atribui automaticamente CWEs às vulnerabilidades, essa associação foi feita manualmente, com base na descrição da falha.

3.4 Análise e Tratamento dos Dados

Os dados foram organizados em planilhas e agrupados por:

- Quantidade total de vulnerabilidades por ferramenta;
- Vulnerabilidades únicas (sem repetições);
- Distribuição por nível de severidade;
- Classificação por CWE.

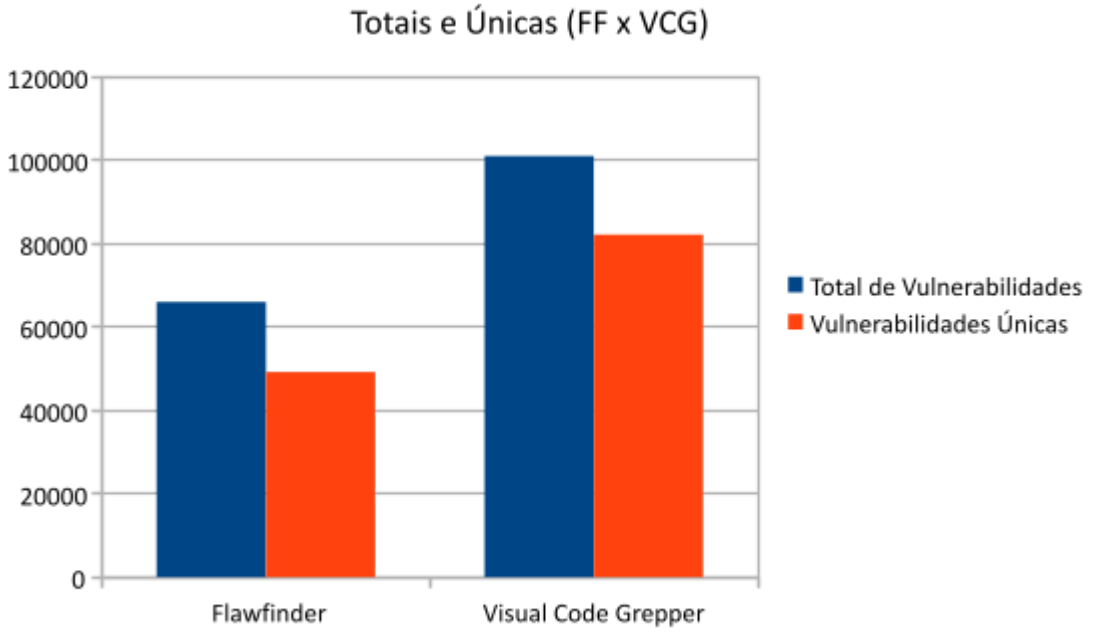
A comparação foi feita tanto quantitativamente, considerando o volume de falhas detectadas, quanto qualitativamente, observando quais CWEs são mais bem identificadas por cada ferramenta.

4 ANÁLISE E DISCUSSÃO

Nesta seção, apresentam-se os dados coletados e os resultados obtidos ao longo da pesquisa. A avaliação se concentrou na quantidade, qualidade e especificidade das vulnerabilidades detectadas por cada ferramenta SAST, com foco nos CWEs identificados. As análises a seguir estão acompanhadas de tabelas comparativas, permitindo uma visualização clara e objetiva dos dados.

4.1 Total de Vulnerabilidades Detectadas

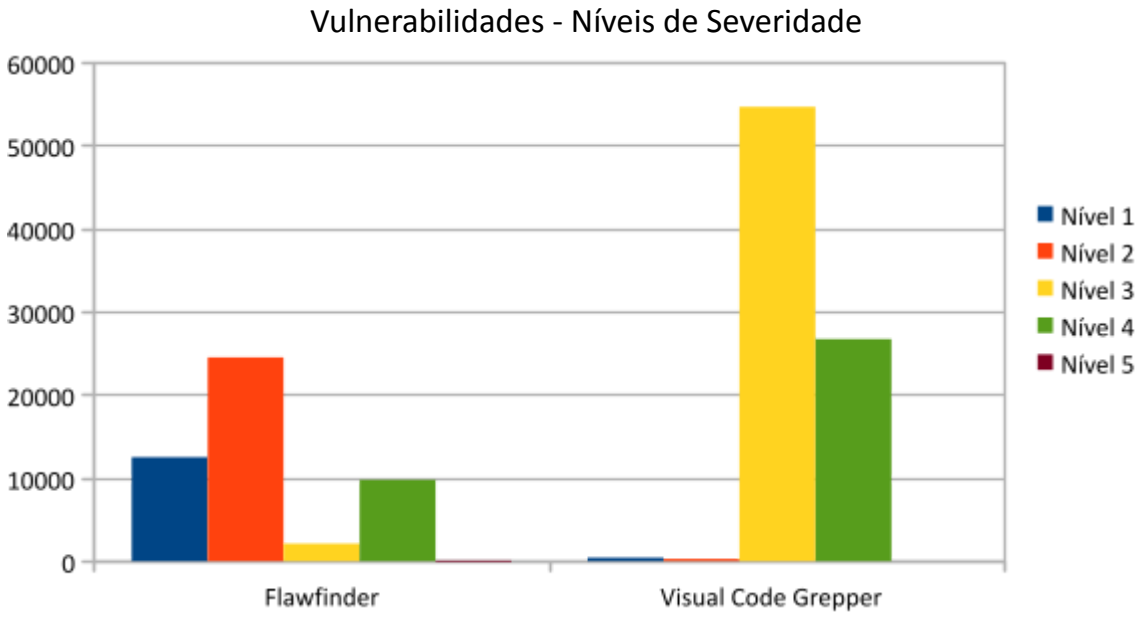
Gráfico 1 – Total de vulnerabilidades detectadas por ferramenta



A ferramenta Visual Code Grepper apresentou um volume maior de vulnerabilidades detectadas, tanto no total quanto em ocorrências únicas, indicando maior sensibilidade em sua varredura. Contudo, isso não necessariamente implica em maior qualidade de detecção.

4.2 Distribuição por Nível de Severidade

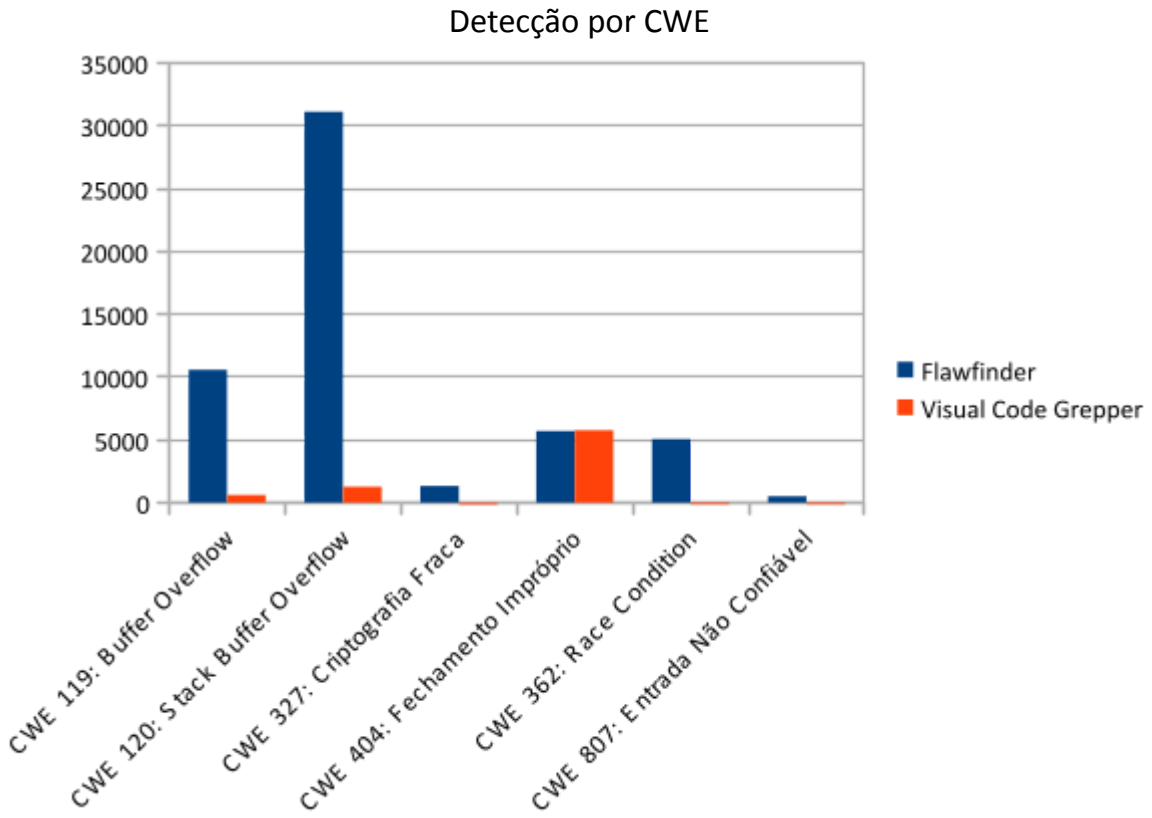
Gráfico 2 – Vulnerabilidades únicas por nível de severidade



Os dados revelam que o VCG tende a identificar mais falhas em níveis altos de severidade (3 e 4), enquanto o Flawfinder detecta um volume maior em níveis mais baixos. Isso indica que o VCG é mais incisivo na identificação de vulnerabilidades críticas em aplicações de alto nível — por exemplo, apontando acessos a arquivos sensíveis sem validação adequada ou uso inseguro de funções responsáveis por operações privilegiadas, que podem resultar em escalonamento de privilégios. Já o Flawfinder mostra maior eficiência em auditorias mais abrangentes de códigos de infraestrutura, frequentemente identificando uso de funções inseguras como strcpy() ou gets(), típicas de vulnerabilidades de menor severidade, mas que ainda representam riscos relevantes quando acumuladas em grandes bases de código. Esses exemplos reforçam a complementaridade entre as ferramentas no processo de análise.

4.3 Detecção por Tipo de Vulnerabilidade (CWE)

Gráfico 3 – Detecção por CWE (exemplos selecionados)



A comparação por CWE evidencia a especialização de cada ferramenta. O Flawfinder se destaca em vulnerabilidades relacionadas a memória e execução de baixo nível, enquanto o VCG apresenta resultados competitivos em falhas relacionadas à manipulação de arquivos e recursos do sistema.

4.4 Discussão Geral

Os resultados demonstram que nenhuma das ferramentas se sobressai em todos os aspectos. Enquanto o Flawfinder possui maior abrangência em CWEs críticos e mais relevantes para sistemas embarcados e de infraestrutura, o VCG é mais eficaz na detecção de vulnerabilidades severas em aplicações voltadas ao usuário.

Em contextos de gestão pública, onde coexistem sistemas legados, web services e plataformas integradas, a utilização combinada das duas ferramentas

representa uma abordagem robusta e abrangente para a prevenção de falhas de segurança.

Em contextos de gestão pública, onde coexistem sistemas legados, web services e plataformas integradas, a utilização combinada das duas ferramentas representa uma abordagem robusta e abrangente para a prevenção de falhas de segurança. Um exemplo concreto ocorre em ambientes de secretarias de saúde municipais, que frequentemente operam um sistema legado de prontuário eletrônico desenvolvido em C/C++ e, ao mesmo tempo, integram esse sistema a plataformas web de agendamento e a serviços nacionais como o Conect SUS. Nesse cenário, vulnerabilidades presentes no código legado como problemas de manipulação de memória podem se propagar para APIs expostas externamente, ampliando o risco de violações de dados sensíveis. A aplicação simultânea das duas ferramentas SAST permite identificar falhas específicas desses diferentes componentes, oferecendo maior cobertura e reduzindo significativamente a probabilidade de exploração.

5 CONSIDERAÇÕES FINAIS

Este estudo teve como objetivo principal avaliar a eficácia de ferramentas SAST open-source e standalone, especificamente o Flawfinder e o Visual Code Grepper (VCG), na detecção de tipos específicos de vulnerabilidades em projetos escritos em C/C++. A pesquisa foi motivada pelo crescente uso de sistemas digitais na gestão pública e pela necessidade de fortalecer a segurança de aplicações críticas frente ao aumento dos crimes e golpes virtuais. Os resultados obtidos confirmaram essa hipótese inicial: ambos os analisadores identificaram um conjunto relevante de vulnerabilidades relacionadas à manipulação insegura de memória e uso de funções depreciadas, com o Flawfinder apresentando maior sensibilidade a falhas de buffer overflow e o VCG destacando-se na detecção de padrões inseguros em trechos de lógica de controle. Esses achados reforçam que

ferramentas SAST acessíveis podem contribuir significativamente para elevar o nível de segurança em sistemas públicos.

A análise permitiu constatar que ambas as ferramentas possuem capacidades relevantes, mas distintas: o Flawfinder demonstrou maior eficácia na detecção de vulnerabilidades relacionadas à manipulação de memória e execução de baixo nível (como CWE-119 e CWE-120), identificando cerca de 65% mais ocorrências desse tipo de falha em comparação ao VCG. Por outro lado, o VCG destacou-se na identificação de falhas relacionadas ao uso impróprio de recursos e à manipulação de arquivos (como CWE-404), encontrando aproximadamente o dobro de alertas desse grupo em relação ao Flawfinder. Esse contraste quantitativo reforça que as ferramentas se complementam ao cobrir diferentes categorias de vulnerabilidades.

Os resultados indicam que os objetivos da pesquisa foram alcançados, permitindo identificar pontos fortes e limitações de cada ferramenta. A hipótese de que o uso combinado dessas soluções poderia oferecer uma cobertura mais ampla e complementar foi confirmada. A pesquisa também evidenciou que ferramentas SAST gratuitas e executáveis localmente podem atender a requisitos de segurança, privacidade e controle de ambientes governamentais, o que é essencial em contextos de gestão pública.

Entre as limitações do estudo, destaca-se a restrição à linguagem C/C++ e ao uso de apenas duas ferramentas. Além disso, a associação manual das vulnerabilidades detectadas pelo VCG às CWEs pode ter introduzido viés de interpretação, apesar do rigor aplicado nesse processo. Essas limitações implicam que os resultados não podem ser generalizados para outras linguagens, ferramentas ou contextos de análise, e que parte das classificações pode refletir interpretações do pesquisador. Assim, embora os achados sejam relevantes, eles devem ser compreendidos dentro desse escopo específico e considerados como ponto de partida para investigações mais amplas.

Como possibilidades futuras de pesquisa, recomenda-se expandir o escopo da análise para outras linguagens de programação amplamente utilizadas na esfera pública, como Java, Python e PHP, bem como incluir ferramentas SAST adicionais, além de métodos complementares como DAST e IAST. Outra vertente promissora é a avaliação do impacto da integração dessas ferramentas em pipelines DevSecOps, especialmente em instituições públicas que buscam maturidade em segurança de software.

REFERÊNCIAS

AHMED, A. (2019). **Devsecops: Enabling security by design in rapid software development. Master's thesis**.

AL BREIKI, Hamda & Mahmoud, Qusay. (2014). **Evaluation of static analysis tools for software security**. 93-98. 10.1109/INNOVATIONS.2014.6987569.

BACA, D., Carlsson, B., and Lundberg, L. (2008). **Evaluating the cost reduction of static code analysis for software security**. In Proceedings of the third ACM SIGPLAN workshop on Programming languages and analysis for security, pages 79–88.

DIAS, R. d. C. W. B. et al. (2023). **Avaliação comparativa das metodologias na gestão de projetos**.

ESPOSITO, M., Falaschi, V., and Falessi, D. (2024). **An extensive comparison of static application security testing tools.**

FARRAPO, Valdeclébio et al. **Evaluating the Use of Open-Source and Standalone SAST Tools for Detecting Vulnerabilities in C/C++ Projects**. In Proceedings of the 27th International Conference on Enterprise Information Systems - Volume 1: ICEIS; ISBN 978989-758-749-8, SciTePress, pages 394-401. DOI: 10.5220/0013483500003929.

MITRE CORPORATION. CWE – Common Weakness Enumeration. Disponível em: https://cwe.mitre.org. Acesso em: 11 ago. 2025.

MITRE CORPORATION. CVE – Common Vulnerabilities and Exposures. Disponível em: https://cve.mitre.org. Acesso em: 11 ago. 2025.

OPEN WEB APPLICATION SECURITY PROJECT. OWASP Top 10: The ten most critical web application security risks. 2021. Disponível em: https://owasp.org/www-project-top-ten/.

Acesso em: 11 ago. 2025.

Wheeler, D. A. (2015). **Secure programming howto. Walters Art Museum in Baltimore, Maryland**.

VISUAL CODE GREPPER. Visual Code Grepper – static code analysis tool. 2023. Disponível em: https://sourceforge.net/projects/visualcodegrepp/. Acesso em: 11 ago. 2025.