

Avaliação de Vulnerabilidades em Nuvem: Um Estudo Comparativo de Scanners de IaC

Léo Walker da Silva¹, Mayza Umbelina Fidelis da Silva¹, Melissa Behrends Rodrigues¹, Marlon Vinícius da Silva¹, Wesley Jean Ferreira Pereira¹

¹Faculdade de Ciências Exatas e Tecnológicas (FACET) – Universidade do Estado do Mato Grosso (UNEMAT). Campus de Cáceres Av. São João, n. 563 – Cavallhada I – CEP: 78.216-060 – Cáceres – MT – Brasil

walker.leo@unemat.br, melissa.rodrigues@unemat.br,
mayza.silva@unemat.br, marlon.silva@unemat.br,
wesley.pereira@unemat.br

Abstract. *Infrastructure as Code (IaC) cloud provisioning increases the risk of replicating insecure configurations. This study compares three open-source Static Application Security Testing (SAST) tools—Checkov, Trivy, and tfsec—using the TerraGoat benchmark. Criteria included detection rate, execution time, and usability. Results show Checkov has the widest detection range (467 anomalies), while Trivy offers the best performance for Continuous Integration and Continuous Deployment (CI/CD) automation, executing in 1 second. The choice depends on specific DevSecOps requirements and infrastructure maturity.*

Resumo. *A Infraestrutura como Código (IaC) otimizou o provisionamento mas ampliou a replicação de configurações inseguras. Este estudo compara três ferramentas de Análise Estática de Segurança de Código (SAST — Static Application Security Testing) de código aberto: Checkov, Trivy e tfsec, via benchmark TerraGoat. Os critérios incluíram taxa de detecção, tempo de execução e usabilidade. O Checkov obteve maior abrangência (467 anomalias), enquanto o Trivy apresentou melhor desempenho para automação de Integração Contínua e Entrega Contínua (CI/CD), com execução em 1 segundo. A escolha depende da maturidade DevSecOps.*

1. Introdução

A evolução da computação em nuvem e a consolidação da cultura *DevOps* transformaram a maneira como infraestruturas são provisionadas. O paradigma de Infraestrutura como Código (IaC) permite que servidores, redes e bancos de dados sejam definidos através de arquivos de configuração, garantindo escalabilidade, versionamento e repetibilidade. No entanto, essa agilidade introduz um novo vetor de risco: configurações inseguras escritas em código (como *buckets* S3 públicos ou chaves de acesso expostas) podem ser replicadas em larga escala em poucos segundos.

Nesse cenário, a segurança “*shift-left*”, a prática de mover os testes de segurança para as fases iniciais do ciclo de desenvolvimento torna-se indispensável. As ferramentas de Análise Estática de Segurança de Infraestrutura (*IaC Scanning*) surgem como a primeira linha de defesa, permitindo que vulnerabilidades sejam

detectadas nos arquivos de configuração antes mesmo do provisionamento dos recursos no ambiente de nuvem.

Apesar da abundância de soluções disponíveis no mercado, existe uma lacuna de comparação prática sobre a eficiência e a verbosidade dessas ferramentas quando aplicadas a ambientes intencionalmente vulneráveis. Qual ferramenta oferece o melhor equilíbrio entre tempo de execução e precisão na detecção de falhas críticas? Como a saída de dados de cada uma influencia a tomada de decisão do desenvolvedor?

Este artigo apresenta um estudo comparativo entre três das principais ferramentas de código aberto do mercado: Checkov (BRIDGECREW, 2026), Trivy (AQUA SECURITY, 2026) e tfsec (AQUA SECURITY, 2026). Cabe destacar que o *tfsec* encontra-se em processo de migração e consolidação para dentro do motor do Trivy pela sua desenvolvedora (Aqua Security); contudo, sua avaliação isolada no laboratório TerraGoat mantém-se válida e necessária para a compreensão empírica da evolução dos motores de análise. O objetivo é fornecer aos engenheiros de segurança critérios claros para a escolha da solução mais adequada às esteiras de Integração e Entrega Contínuas (CI/CD).

2. Fundamentação Teórica

Para compreender a relevância da análise de vulnerabilidades em ambientes de nuvem, é necessário estabelecer os conceitos de Infraestrutura como Código (IaC), o paradigma de segurança moderna e o estado da arte na literatura recente.

2.1 Infraestrutura como Código (IaC) e Terraform

A Infraestrutura como Código (IaC) é o processo de gerir e provisionar infraestruturas de TI através de ficheiros de definição legíveis por máquina, em vez de configurações manuais. O *Terraform*, desenvolvido pela HashiCorp, consolidou-se como a ferramenta líder de mercado por ser agnóstica em relação ao provedor de nuvem. Utilizando a linguagem declarativa *HashiCorp Configuration Language* (HCL), permite que o estado desejado da infraestrutura seja versionado e auditado.

Contudo, a facilidade de automatização traz riscos proporcionais. Segundo War et al. (2025), a proliferação de vulnerabilidades em scripts de IaC é um desafio crescente, pois uma única falha de configuração num ficheiro base pode propagar-se por centenas de recursos na nuvem, expondo dados sensíveis antes mesmo do sistema entrar em fase de operação.

2.2 DevSecOps e o Paradigma "Shift-Left"

O modelo *DevSecOps* surge para integrar a segurança de forma contínua no ciclo de vida do desenvolvimento. Dentro deste contexto, o conceito de *Shift-Left Security* propõe que os testes de segurança sejam antecipados para as fases iniciais do projeto. Conforme destacado por Singh et al. (2024), a utilização de ferramentas de análise de IaC é uma estratégia vital para fortalecer a segurança em ambientes de *Cloud DevSecOps*, permitindo que a detecção de erros ocorra durante a escrita do código, reduzindo drasticamente o custo e a complexidade de mitigação em comparação com correções feitas em ambiente de produção.

2.3 Trabalhos Relacionados

A eficácia das ferramentas de análise estática (SAST) tem sido objeto de estudos rigorosos. Verdet et al. (2025) realizaram uma avaliação detalhada sobre a adoção de políticas de segurança por parte de desenvolvedores em projetos *Terraform*, utilizando scanners como *Checkov* e *tfsec*. O estudo revelou que, embora existam ferramentas robustas, a eficácia da proteção depende da integração correta destas soluções no fluxo de trabalho e da capacidade das ferramentas em identificar configurações inseguras em diferentes provedores de nuvem.

Complementarmente, a investigação de War et al. (2025) reforça que a análise automática de scripts de IaC não é apenas uma recomendação, mas uma necessidade empírica, dado o volume de falhas críticas encontradas em repositórios públicos. O presente artigo alinha-se a estas investigações ao propor uma comparação prática entre o *Checkov*, o *Trivy* e o *tfsec*, focando-se na sua capacidade de resposta perante o laboratório de vulnerabilidades controladas *TerraGoat*.

3. Metodologia

Esta pesquisa adota uma abordagem experimental e quantitativa, baseada na execução controlada de ferramentas de Análise Estática de Segurança de Código (SAST) contra um ambiente de testes padronizado. O objetivo é comparar a eficácia de diferentes soluções de código aberto na identificação de vulnerabilidades em scripts de Infraestrutura como Código (IaC).

3.1 Ambiente de Teste (Laboratório)

Para garantir a imparcialidade e a reprodutibilidade do experimento, optou-se por não utilizar códigos proprietários. Em vez disso, os testes foram conduzidos sobre o *TerraGoat* (mantido pela Bridgecrew/Palo Alto Networks), um projeto de código aberto amplamente reconhecido na comunidade de cibersegurança. O *TerraGoat* atua como um laboratório de aprendizado contendo dezenas de arquivos de configuração do *Terraform* intencionalmente vulneráveis, abrangendo falhas em provedores como *AWS*, *Azure* e *Google Cloud Platform (GCP)*.

3.2 Ferramentas Avaliadas e Métricas de Comparação

Para avaliar o desempenho e a aplicabilidade de cada ferramenta, os resultados foram tabulados com base em três métricas principais, escolhidas por refletirem os requisitos de segurança e operacionalidade:

Taxa de Detecção (Cobertura): Mede a eficácia na identificação de riscos (indicador de "quanto maior, melhor"). Esta métrica valida a capacidade do motor em descobrir e categorizar falhas por nível de severidade.

Tempo de Execução (Desempenho): O tempo total gasto para concluir a varredura completa (indicador de "quanto menor, melhor"). A métrica justifica-se pelo requisito de agilidade do modelo DevOps, onde análises lentas tornam-se gargalos na esteira.

Usabilidade do Relatório: Avaliação qualitativa da clareza das saídas para redução do *Mean Time To Repair* (MTTR), avaliando se a ferramenta fornece a localização exata do código vulnerável e links de correção.

3.3 Procedimento Experimental

O experimento foi realizado em um ambiente local controlado para garantir que as métricas de tempo não sofressem interferência de latência de rede. Para a execução dos testes, utilizou-se o Windows Subsystem for Linux (WSL2) rodando sobre o sistema operacional Windows 10 (versão 10.0.19045). O ambiente de execução foi a distribuição Ubuntu 24.04.2 LTS (Noble Numbat), operando com a versão do kernel Linux 6.6.87.2-1 e versão do WSL 2.6.3.0. O procedimento seguiu as seguintes etapas:

1. Clonagem do repositório oficial do projeto *TerraGoat* para o ambiente local.
2. Instalação das versões mais recentes (estáveis) do *Checkov*, *Trivy* e *tfsec*.
3. Execução individual de cada ferramenta apontando para o diretório raiz do *TerraGoat*, utilizando as configurações padrão (*default*) de varredura, sem a aplicação de filtros ou regras customizadas.
4. Extração e armazenamento dos relatórios gerados por cada varredura.

4. Resultados e Discussão

Para avaliar a eficácia e o desempenho das ferramentas propostas, o repositório *TerraGoat* foi submetido a varreduras padronizadas utilizando o *Checkov*, o *tfsec* e o *Trivy*, executados em contêineres Docker (plataforma de virtualização de nível de sistema operacional), isolados para garantir a imparcialidade do ambiente. Os critérios de avaliação priorizaram o volume de vulnerabilidades detectadas, a severidade apontada, o tempo de execução da análise e a usabilidade do formato de saída. Os resultados consolidados estão apresentados na Tabela 1.

Tabela 1: Desempenho Comparativo dos Scanners de IaC

Ferramenta	Total de Falhas	Críticas/Altas	Tempo	Formato de Saída
Checkov	467	182	13s	Terminal Detalhado
tfsec	246	109	3s	Terminal Focado
Trivy	132	63	1s	Terminal em Tabela

Fonte: Elaborado pelos autores (2026)

A análise dos dados revela que o *Checkov* possui a maior abrangência absoluta, detectando 467 anomalias. Esse volume superior decorre de sua

arquitetura baseada em grafos, que permite escaneamentos de segredos (*secrets*) e arquivos de suporte (como Dockerfiles). No entanto, essa abrangência sistêmica gera um número expressivo de alertas puramente informacionais e potenciais falsos positivos, justificando a discrepância no volume total de falhas reportadas. Ademais, essa profundidade resulta em um tempo de execução de 13 segundos e em uma saída altamente verbosa, dificultando a triagem em cenários de desenvolvimento ágil.

O *tfsec* demonstrou um rigor notável na categorização de riscos específicos para Terraform, apontando 109 falhas críticas e altas em apenas 3 segundos. Conforme documentado no repositório oficial da ferramenta, a equipe mantenedora iniciou a consolidação de seus esforços de varredura diretamente no Trivy, visando fornecer uma solução corporativa unificada. Mesmo com a migração em andamento, o *tfsec* apresentou resultados sólidos para revisões de segurança granulares.

Por fim, o Trivy apresentou o melhor equilíbrio prático para a automação contínua. Com o menor tempo de execução (1 segundo) e estruturação em formato tabular, minimiza a fadiga de alertas. A identificação de 132 vulnerabilidades indica uma abordagem opinativa, focada em falhas efetivamente exploráveis em detrimento de avisos de conformidade, o que favorece a agilidade em esteiras DevSecOps.

5. Conclusão

A adoção do modelo de Infraestrutura como Código (IaC) revolucionou a agilidade na implantação de serviços em nuvem, mas transferiu a responsabilidade pela configuração de segurança diretamente para as fases iniciais de desenvolvimento. Este artigo cumpriu seu objetivo ao realizar uma avaliação prática das ferramentas de análise estática *Checkov*, *Trivy* e *tfsec*, utilizando o ambiente vulnerável padrão *TerraGoat* como *benchmark*.

Os resultados demonstraram que não existe uma única ferramenta universal, mas sim soluções adequadas a diferentes níveis de maturidade nas esteiras DevSecOps. O *Checkov* destacou-se pela abrangência absoluta, detectando 467 anomalias por meio de sua análise baseada em grafos, o que o torna ideal para auditorias profundas de conformidade, apesar do maior tempo de execução (13s) e alta verbosidade. Em contrapartida, o *tfsec* apresentou rigor notável em arquivos Terraform, identificando 109 falhas críticas e altas em apenas 3 segundos.

No entanto, o *Trivy* consolidou-se como o melhor equilíbrio prático para a automação contínua. Com execução em 1 segundo e estruturação de relatórios em formato tabular, ele minimiza a fadiga de alertas e facilita a remediação imediata por parte dos desenvolvedores. Um dado estratégico revelado durante este estudo é o processo de fusão do motor do *tfsec* ao *Trivy* sob a gestão da Aqua Security; essa movimentação de mercado sinaliza uma forte tendência de consolidação em binários unificados para reduzir a complexidade de manutenção das esteiras de CI/CD.

Para trabalhos futuros, recomenda-se a expansão deste estudo prático para quantificar a taxa de falsos positivos gerados por cada ferramenta e avaliar o desempenho dos scanners perante a injeção de políticas customizadas (*custom*

policies), fundamentais para atender aos requisitos específicos de governança corporativa.

6. Referências Bibliográficas

- AQUA SECURITY. tfsec: Security scanner for Terraform. Disponível em: <https://aquasecurity.github.io/tfsec/>. Acesso em: 8 abr. 2026.
- AQUA SECURITY. Trivy: Comprehensive and versatile security scanner. Disponível em: <https://trivy.dev>. Acesso em: 8 abr. 2026.
- BRIDGECREW. Checkov: Static code analysis for infrastructure as code. Disponível em: <https://www.checkov.io>. Acesso em: 8 abr. 2026.
- SINGH, R.; YEBOAH-OFORI, A.; KUMAR, S.; GANIYU, A. Fortifying cloud DevSecOps security using Terraform infrastructure as code analysis tools. In: INTERNATIONAL CONFERENCE ON ELECTRICAL AND COMPUTER ENGINEERING RESEARCHES (ICECER), 2024, Gaborone. Proceedings [...]. IEEE, 2024. p. 1–6. DOI: 10.1109/ICECER62944.2024.10920371.
- VERDET, A.; HAMDAQA, M.; DA SILVA, L.; KHOMH, F. Assessing the adoption of security policies by developers in Terraform across different cloud providers. *Empirical Software Engineering*, v. 30, n. 3, 2025.
- WAR, A.; HABIB, A.; DIALLO, A.; KLEIN, J.; BISSYANDÉ, T. F. Vulnerabilities in infrastructure as code: what, how many, and who? *Empirical Software Engineering*, v. 30, 2025.