Uma Abordagem DevSecOps para Inserção e Automação de Práticas de Segurança em Pipelines CI/CD

Guilherme Henrique de Lima Machado

[1]Instituto de Ciências Exatas e Informática – Pontíficia Universidade Católica de Minas Gerais
Minas Gerais (PUC Minas)
Contagem – MG – Brasil

guilhermeoh.machado@gmail.com

Resumo. *A integração da segurança no ciclo de vida de desenvolvimento de software, ou DevSecOps, tornou-se essencial para aumentar a confiabilidade e reduzir riscos nas entregas contínuas de software. No entanto, muitas organizações ainda tratam a segurança de forma tardia, existindo uma lacuna sobre como incorporar práticas de segurança de forma contínua e automatizada em pipelines CI/CD. O objetivo geral deste trabalho é investigar uma abordagem prática para a inserção e automação de práticas de segurança em pipelines CI/CD, alinhadas aos princípios de DevSecOps. Para isso, foi realizada uma pesquisa experimental que envolveu o provisionamento de um ambiente controlado na Google Cloud Platform (GCP) utilizando Terraform. Um pipeline no Cloud Build foi configurado para executar um conjunto de ferramentas de segurança, incluindo SAST (Semgrep), DAST (OWASP ZAP), SCA (OWASP Dependency-Check) e IaC Scan (Checkov), contra uma aplicação intencionalmente vulnerável (DVWA). O trabalho conclui com uma análise qualitativa da efetividade da integração, do nível de automação alcançado e das boas práticas observadas.*

1. Introdução

A indústria de software adotou amplamente metodologias ágeis e a cultura DevOps para acelerar a entrega de valor ao cliente. No centro dessa transformação está o pipeline de **Integração Contínua e Entrega Contínua (CI/CD)**, um processo automatizado que gerencia as etapas de build, teste e implantação. Essa automação permite que organizações liberem novas versões de software em um ritmo sem precedentes, respondendo rapidamente às demandas do mercado.

Contudo, essa velocidade expôs uma falha crítica nos processos legados: as práticas de segurança tradicionais não conseguem acompanhar o rápido Ciclo de Vida de Desenvolvimento de Software (SDLC) [Rangnau et al. 2020]. Historicamente, a segurança era tratada como uma fase final, manual e executada por equipes isoladas [Ahmed and Francis 2019]. Isso funciona como um gargalo que contradiz a agilidade do DevOps, fazendo com que a segurança seja percebida como um obstáculo que desacelera os processos [Santos et al. 2024]. Como resultado, muitas equipes negligenciam a segurança ou a aplicam tardiamente, o que gera vulnerabilidades, retrabalho e aumenta o risco em produção.

Para resolver esse conflito, surgiu a cultura **DevSecOps** (Desenvolvimento, Segurança e Operações). A sua definição consiste em uma abordagem que integra controles e processos de segurança ao ciclo de vida DevOps, com colaboração entre as

equipes de segurança, desenvolvimento e operações [Putra and Kabetta 2022]. O DevSecOps torna a equipe inteiramente responsável pela segurança da aplicação, implementando decisões de segurança na mesma velocidade que as tarefas de desenvolvimento [Ahmed and Francis 2019, Kushwaha et al. 2024]. A principal estratégia adotada é a de **"shift-left"**, que consiste em introduzir práticas e ferramentas de segurança o mais cedo possível no processo [Chen and Suo 2022], de forma automatizada dentro do próprio pipeline CI/CD.

Apesar do consenso sobre a importância do DevSecOps, o problema de *como* incorporar efetivamente as práticas de segurança de forma contínua e automatizada persiste. A literatura recente apresenta estudos de caso práticos, muitos focando na integração de ferramentas específicas como SAST (Static Application Security Testing) e DAST (Dynamic Application Security Testing) [Putra and Kabetta 2022, Marandi et al. 2023, Kushwaha et al. 2024, Sun et al. 2021], ou aprofundando-se nos desafios técnicos de uma única técnica, como o DAST [Rangnau et al. 2020]. No entanto, ainda há uma lacuna em abordagens práticas que demonstrem a orquestração de um *conjunto* mais amplo de ferramentas (incluindo SAST, DAST, SCA e IaC Scan) de forma unificada e reprodutível em ambientes de nuvem modernos [Bernardino et al. 2024].

Este trabalho busca preencher essa lacuna, tendo como objetivo geral investigar uma abordagem prática para a inserção e automação de práticas de segurança em pipelines CI/CD. Para isso, adota-se uma pesquisa experimental. Foi provisionado um ambiente controlado na Google Cloud Platform (GCP), utilizando Terraform para garantir a reprodutibilidade da infraestrutura (IaC). Um pipeline no Cloud Build foi configurado para orquestrar um conjunto de ferramentas de segurança de código aberto contra a aplicação-alvo DVWA (*Damn Vulnerable Web Application*). Os objetivos específicos incluem aliar estratégias de automação como SAST (Semgrep), DAST (OWASP ZAP), SCA (OWASP Dependency-Check) e IaC scanning (Checkov), validando a efetividade da integração.

Este artigo está estruturado da seguinte forma: a Seção 2 apresenta a fundamentação teórica sobre os conceitos de DevSecOps, Teste Contínuo de Segurança e pipelines CI/CD. A Seção 3 discute os trabalhos relacionados que fundamentam esta pesquisa. A Seção 4 detalha os materiais e métodos empregados na pesquisa experimental. A Seção 5 apresenta e discute os resultados obtidos. Finalmente, a Seção 6 apresenta as conclusões e sugestões de trabalhos futuros.

2. Fundamentação Teórica

Esta seção aborda os três pilares conceituais que sustentam esta pesquisa: a cultura DevSecOps, o processo de Teste Contínuo de Segurança e a plataforma tecnológica do Pipeline CI/CD.

O conceito principal é o **DevSecOps**, uma evolução cultural do DevOps que integra formalmente a segurança ao processo. Sua definição vai além de ferramentas, sendo uma combinação de desenvolvimento, segurança e operações que torna toda a equipe responsável pela segurança da aplicação [Ahmed and Francis 2019, Chen and Suo 2022]. Isso é alcançado ao implementar atividades e decisões de segurança na mesma medida e velocidade que as tarefas de desenvolvimento e operações. O objetivo é introduzir a segurança mais cedo no ciclo de vida (shift-left), em vez de tratá-la como uma etapa final e isolada [Kushwaha et al. 2024], quebrando os silos tradicionais entre as equipes

[Santos et al. 2024, Efendi et al. 2021].

Para implementar o DevSecOps na prática, adota-se o **Continuous Security Testing (Teste Contínuo de Segurança)**. Este conceito é definido como o princípio de aplicar o teste de segurança do software e seus componentes de forma contínua ao longo de todo o pipeline [Santos et al. 2024, Bernardino et al. 2024]. Em vez de uma única auditoria de segurança antes do lançamento, o teste contínuo envolve a execução automatizada de diferentes tipos de análise em fases distintas, como SAST (white-box) e DAST (black-box) [Masood and Java 2015]. Isso permite que as equipes de segurança acompanhem a velocidade do DevOps e forneçam feedback rápido aos desenvolvedores, garantindo que vulnerabilidades sejam detectadas e corrigidas proativamente [Rangnau et al. 2020].

O alicerce tecnológico que permite a automação tanto do DevSecOps quanto do teste contínuo é o **Pipeline CI/CD**. Este é o componente central do DevSecOps, composto por um conjunto de fases consecutivas (como Plan, Code, Build, Test, Deploy) que definem atividades e ferramentas para auxiliar na automação do desenvolvimento [Santos et al. 2024, Sun et al. 2021]. Um pipeline moderno, frequentemente executado sobre uma infraestrutura responsiva baseada em contêineres (Docker, Kubernetes) [Díaz et al. 2019], orquestra a execução de ferramentas de segurança. Ele trata as falhas de segurança da mesma forma que falhas de build ou testes funcionais: interrompendo o pipeline e exigindo correção antes da promoção para o próximo estágio [Marandi et al. 2023, Putra and Kabetta 2022].

3. Trabalhos Relacionados

A literatura sobre DevSecOps tem crescido, com foco em frameworks, estudos de caso e desafios de implementação. Diversos autores propõem frameworks para guiar a adoção do DevSecOps. [Santos et al. 2024] apresenta um dos trabalhos mais alinhados a esta pesquisa, propondo um framework de oito fases e aplicando-o em um estudo de caso real (Projeto GRACE) que integra SAST, DAST e SCA. De forma complementar, [Chen and Suo 2022] projeta uma arquitetura de segurança de 10 fases focada na automação e [Efendi et al. 2021] estuda a transformação cultural de uma organização, propondo um modelo de 3 fases (Mindset, Práticas e Otimização) para sair do modelo Waterfall. Esta pesquisa se baseia nesses frameworks para propor uma implementação prática focada na orquestração de ferramentas.

Uma vertente significativa da literatura foca em estudos de caso que implementam a combinação de SAST e DAST. [Putra and Kabetta 2022] apresenta uma implementação prática que integra SAST e DAST em um pipeline GitLab/Docker, medindo a redução no tempo de deploy. Similarmente, [Marandi et al. 2023] foca na automação de SAST (Snyk) e DAST (StackHawk) em GitHub Actions, destacando o uso de dashboards. [Kushwaha et al. 2024] também detalha uma implementação em ambiente financeiro usando Azure DevOps, integrando SAST (Codacy), DAST (OWASP ZAP) e GHAS. Esses trabalhos validam a viabilidade de orquestrar SAST e DAST, um objetivo central desta pesquisa.

Enquanto os trabalhos anteriores combinam técnicas, outros se aprofundam nos desafios de tipos de testes específicos. [Rangnau et al. 2020] oferece uma análise fundamental sobre os desafios técnicos de integrar DAST, focando no *overhead* (tempo de execução) e na complexidade de automação de ferramentas como o OWASP ZAP, também

usado neste trabalho. No outro extremo, [Masood and Java 2015] fornece uma análise teórica profunda sobre SAST, comparando as abordagens white-box (SAST) e black-box (DAST) e justificando a necessidade de ambas, observando que a cobertura do SAST é tipicamente maior que a dos testes de penetração.

Outros trabalhos focam na arquitetura de pipeline e na infraestrutura de automação. [Sun et al. 2021] projeta e implementa um pipeline de teste de segurança que visa integrar-se perfeitamente ao CI/CD (usando Jenkins/GitLab) e unificar os resultados de múltiplas ferramentas (Sonar, Fortify, Appscan) em uma plataforma de gerenciamento centralizada. Por sua vez, [Díaz et al. 2019] foca na infraestrutura responsiva necessária para suportar processos DevSecOps de alta disponibilidade, defendendo o uso de contêineres (Docker) e orquestradores (Kubernetes) como base tecnológica para a automação. Esta pesquisa combina ambas as visões, usando um orquestrador (Cloud Build) para executar as ferramentas sobre uma infraestrutura baseada em contêineres (GKE).

Finalmente, a literatura aborda os desafios culturais e as lacunas de ferramentas. [Ahmed and Francis 2019] fornece o contexto cultural, discutindo os desafios de processo e mentalidade ao introduzir a segurança em um fluxo DevOps existente. [Bernardino et al. 2024] identifica desafios práticos, como a complexidade de integração de ferramentas e a dificuldade de automatizar tarefas manuais. Para resolver essas lacunas, [Bernardino et al. 2024] propõe o desenvolvimento de ferramentas customizadas (para integração com JIRA e checagem de versões de componentes), reforçando a necessidade de abordagens práticas, como a deste artigo, para preencher as lacunas que as ferramentas de mercado deixam.

4. Materiais e Métodos

Para atingir os objetivos propostos, foi definida uma metodologia de **pesquisa experimental**. Este tipo de pesquisa é justificado pelo desenvolvimento de um ambiente controlado e instrumentado, onde variáveis podem ser gerenciadas para observar seus efeitos. O experimento consiste em provisionar uma infraestrutura de nuvem, implementar um pipeline CI/CD instrumentado com ferramentas de segurança e executar este pipeline contra uma aplicação-alvo intencionalmente vulnerável. A análise dos resultados gerados permite validar a abordagem de automação.

O ambiente experimental foi construído inteiramente na **Google Cloud Platform (GCP)**. A plataforma foi escolhida por seus serviços gerenciados que facilitam a automação. O **Cloud Build** foi utilizado como o orquestrador do pipeline CI/CD. O **Google Kubernetes Engine (GKE)** serviu como o ambiente de destino para a implantação da aplicação. Para o armazenamento de imagens Docker, foi utilizado o **Artifact Registry**, e para o arquivamento dos relatórios de segurança gerados, foi usado o **Cloud Storage (GCS)**.

Uma premissa central do projeto é a **reprodutibilidade**. Toda a infraestrutura de nuvem descrita (VPCs, cluster GKE, repositórios e buckets) foi provisionada de forma declarativa utilizando a ferramenta de Infraestrutura como Código (IaC) **Terraform** (versão ¿= 1.5). Isso garante que o ambiente experimental possa ser recriado de forma consistente, permitindo a replicação dos resultados e a validação do método.

Como aplicação-alvo para os testes de segurança, foi selecionada a **DVWA (*Damn Vulnerable Web Application*)**. O DVWA é uma aplicação web escrita em PHP/MySQL,

amplamente utilizada na comunidade de segurança por ser intencionalmente vulnerável. Seu uso permite avaliar a efetividade das ferramentas de varredura em um ambiente controlado, onde as vulnerabilidades são conhecidas *a priori*. O pipeline utiliza a imagem Docker pública da aplicação (`vulnerables/web-dvwa`).

O núcleo do experimento é o pipeline definido no arquivo `cloudbuild.yaml`, que orquestra um conjunto de ferramentas de segurança de código aberto em etapas distintas. Para **SAST**, foi utilizado o **Semgrep**; para **SCA** (análise de dependências), o **OWASP Dependency-Check**; para **IaC Scan** (varredura dos arquivos Kubernetes), o **Checkov**; para **Container Scan** (varredura da imagem Docker), o **Trivy**; e para **DAST**, o **OWASP ZAP**. Cada ferramenta foi configurada para executar em uma etapa específica do pipeline, antes da implantação final.

A coleta de dados e o método de análise são qualitativos. A execução do pipeline é disparada pelo comando `gcloud builds submit`. Ao final da execução, cada ferramenta de segurança gera um relatório de vulnerabilidades (em formatos `.json` ou `.html`). Esses relatórios são automaticamente carregados e versionados em um bucket do GCS. A análise qualitativa foca nos seguintes pontos: a efetividade da integração das ferramentas, o nível de automação alcançado, as boas práticas observadas e a capacidade de detecção de vulnerabilidades em cada etapa do pipeline.

5. Resultados e Discussões

6. Conclusão

Referências

Ahmed, Z. and Francis, S. C. (2019). Integrating security with devsecops: Techniques and challenges. In *2019 International Conference on Digitization (ICD)*, pages 178–182.

Bernardino, N. A., Sequeira, B., Piza, E., Henriques, F., Neves, F., and Reis, C. I. (2024). Enhancing devsecops: Three custom tools for continuous security. In *2024 IEEE 11th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 53–58.

Chen, T. and Suo, H. (2022). Design and practice of security architecture via devsecops technology. In *2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS)*, pages 310–313.

Díaz, O., Muñoz, M., and Mejía, J. (2019). Responsive infrastructure with cybersecurity for automated high availability devsecops processes. In *2019 8th International Conference On Software Process Improvement (CIMPS)*, pages 1–9.

Efendi, M., Raharjo, T., and Suhanto, A. (2021). Devsecops approach in software development case study: Public company logistic agency. In *2021 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, pages 96–101.

Kushwaha, M. K., David, P., and Suseela, G. (2024). Automation and devsecops: Streamlining security measures in financial system. In *2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–6.

Marandi, M., Bertia, A., and Silas, S. (2023). Implementing and automating security scanning to a devsecops ci/cd pipeline. In *2023 World Conference on Communication & Computing (WCONF)*, pages 1–6.

Masood, A. and Java, J. (2015). Static analysis for web service security - tools & techniques for a secure development life cycle. In *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–6.

Putra, A. M. and Kabetta, H. (2022). Implementation of devsecops by integrating static and dynamic security testing in ci/cd pipelines. In *2022 IEEE International Conference of Computer Science and Information Technology (ICOSNIKOM)*, pages 1–6.

Rangnau, T., v. Buijtenen, R., Fransen, F., and Turkmen, F. (2020). Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines. In *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 145–154.

Santos, N., Escravana, N., Pacheco, B., and Feio, C. (2024). An empirical study of devsecops focused on continuous security testing. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 610–617.

Sun, X., Cheng, Y., Qu, X., and Li, H. (2021). Design and implementation of security test pipeline based on devsecops. In *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, volume 4, pages 532–535.