DAST: Enxergando a Aplicação como um Atacante em Tempo de Execução

Davi de Castro Machado[1], **Ana Lilian**[1]

[1]Universidade Federal de Santa Maria – Santa Maria – Rio Grande do Sul – Brasil

ddmachado@inf.ufsm.br, , ana-emails

Abstract. *Dynamic Application Security Testing (DAST) is a technique that analyzes running applications to identify vulnerabilities from an external perspective, simulating attacks in real-time. This seminar explores DAST concepts, benefits, limitations, and practical integration within DevSecOps pipelines. Popular tools, best practices, and adoption challenges are also discussed, highlighting the importance of dynamic security testing for continuous delivery environments.*

Resumo. *O Dynamic Application Security Testing (DAST) é uma técnica que analisa aplicações em execução para identificar vulnerabilidades sob a perspectiva de um atacante externo, simulando ataques em tempo real. Este seminário apresenta os conceitos, benefícios, limitações e a integração prática do DAST em pipelines DevSecOps. São discutidas ferramentas populares, boas práticas e desafios de adoção, evidenciando a importância dos testes de segurança dinâmicos em ambientes de entrega contínua.*

1. Introdução

A crescente adoção de práticas DevOps trouxe ganhos significativos em agilidade, entrega contínua e automação de processos no ciclo de vida do desenvolvimento de software. No entanto, essa agilidade pode gerar riscos à segurança se esta não for considerada desde o início do processo. Nesse contexto, surge o DevSecOps, que propõe a integração da segurança como uma responsabilidade compartilhada em todas as fases do ciclo de desenvolvimento, desde o planejamento até a produção.

DevSecOps representa a cultura e a prática de incorporar segurança de forma automatizada e contínua, utilizando ferramentas que detectem vulnerabilidades sem comprometer a velocidade do desenvolvimento. Entre essas ferramentas, destaca-se o DAST (Dynamic Application Security Testing), técnica essencial para detecção de falhas em tempo de execução.

O DAST simula ataques contra aplicações em execução, identificando vulnerabilidades exploráveis do ponto de vista de um atacante externo. Trata-se de um complemento importante ao SAST (Static Application Security Testing), por testar a aplicação como um todo, independentemente do código-fonte.

1.1. O que são DAST?

O **DAST** (Dynamic Application Security Testing) é uma técnica de teste de segurança aplicada sobre aplicações em execução. Seu objetivo é simular o comportamento de um usuário malicioso, testando a aplicação da perspectiva externa, como se fosse um invasor.

Ao realizar esse tipo de análise, o DAST é capaz de identificar diversas vulnerabilidades, entre elas:

- Injeção de SQL;
- Cross-site Scripting (XSS);
- Exposição de informações sensíveis;
- Redirecionamentos inseguros.

Por não exigir acesso ao código-fonte, o DAST pode ser utilizado em aplicações legadas ou de terceiros, e é facilmente integrável a ambientes de homologação ou produção. Isso permite avaliações mais próximas do comportamento real da aplicação.

1.2. Relevância do Tema

O uso de DAST é relevante pois permite a detecção de falhas reais, em tempo de execução, replicando cenários que ocorrem em ambientes de produção. Isso o torna uma ferramenta prática e eficaz dentro de pipelines de integração contínua.

No contexto DevSecOps, o DAST garante:

- Feedback rápido sobre riscos de segurança;
- Redução da exposição a ataques;
- Correções proativas antes da entrega ao ambiente produtivo.

Sua capacidade de automatização e sua independência do código-fonte o tornam uma peça-chave para garantir segurança contínua sem comprometer a velocidade do desenvolvimento.

1.3. Objetivos do Trabalho

Este trabalho tem como objetivos principais:

- Apresentar o conceito e funcionamento do DAST;
- Comparar o DAST com outras abordagens de testes de segurança, como SAST e IAST;
- Discutir ferramentas práticas de DAST disponíveis no mercado;
- Analisar os principais desafios e boas práticas na adoção da técnica;
- Relacionar o DAST com os princípios da cultura DevSecOps.

1.4. Revisão Bibliográfica Preliminar

OWASP DEVGUIDE: https://devguide.owasp.org/en/06-verification/02-tools/01-dast/

DevSecOps practices (GitLab, GitHub, Azure DevOps docs)

Artigos como:

[1] [3] [2]

2. Aspectos Teóricos

Os testes de segurança dinâmicos (DAST) se diferenciam de outras abordagens como SAST, IAST e RASP:

- **SAST (Static Application Security Testing)**: análise do código-fonte ou binário sem executar a aplicação.
- **IAST (Interactive Application Security Testing)**: combina testes dinâmicos com instrumentação interna da aplicação.
- **RASP (Runtime Application Self-Protection)**: protege a aplicação em tempo de execução, interceptando ataques enquanto a aplicação está ativa.

2.1. Vantagens do DAST

- Detecta falhas reais e exploráveis em aplicações em execução.
- Não requer acesso ao código-fonte.
- Útil para aplicações legadas ou de terceiros.
- Permite avaliação próxima ao cenário real de uso.

2.2. Limitações do DAST

- Pode gerar falsos positivos.
- Requer um ambiente de teste configurado, similar ao de produção.
- Cobertura limitada sem crawling eficiente da aplicação.
- Pode ser mais lento que SAST devido à execução e interação dinâmica.

3. Aspectos Práticos

Existem diversas ferramentas e frameworks de DAST disponíveis, tanto de código aberto quanto comerciais, que permitem a integração em pipelines de CI/CD e em ambientes de teste.

3.1. Ferramentas DAST populares

- **OWASP ZAP**: ferramenta open-source amplamente utilizada para análise dinâmica de segurança.
- **Burp Suite**: oferece versão gratuita e versão profissional com recursos avançados.
- **Arachni e Nikto**: scanners web que identificam vulnerabilidades comuns.
- **Ferramentas comerciais**: Netsparker, Acunetix, que oferecem relatórios detalhados e integração corporativa.

3.2. Exemplos de uso

- Integração de scans DAST em pipelines CI/CD (GitHub Actions, GitLab CI/CD, Jenkins).
- Escaneamento automatizado de aplicações web após deploy em ambiente de homologação.
- Identificação de vulnerabilidades exploráveis em APIs REST e aplicações web tradicionais.

—¿ Exemplos praticos (mostrar com imagens):

Demonstração de um scan com OWASP ZAP (CLI ou GUI)

Integração de DAST com GitHub Actions ou GitLab CI

4. Desafios e Boas Práticas

A adoção de DAST em ambientes DevSecOps apresenta desafios práticos e técnicos que precisam ser considerados para garantir eficácia e integração nos pipelines de desenvolvimento.

4.1. Desafios

- Ambientes inconsistentes ou não prontos para testes, dificultando a execução confiável dos scans.
- Dificuldade de automatizar 100% do processo, exigindo ajustes e tuning das ferramentas.
- Possibilidade de falsos positivos, que demandam revisão manual.
- Lentidão em scans completos, especialmente em aplicações complexas ou grandes.

4.2. Boas Práticas

- Utilizar DAST em conjunto com SAST, obtendo uma cobertura mais completa de segurança.
- Automatizar os scans com limites de tempo e checkpoints para não impactar o pipeline.
- Executar os testes em ambientes dedicados (staging), evitando impactos na produção.
- Ajustar e configurar corretamente as regras de escaneamento (tuning) para reduzir falsos positivos.
- Avaliar os relatórios em colaboração com times de desenvolvimento e segurança, garantindo que os riscos sejam tratados adequadamente.

5. Conclusão

O DAST é uma ferramenta fundamental para a segurança dinâmica dentro da cultura DevSecOps. Ele complementa abordagens estáticas como SAST, oferecendo visão prática e realista da exposição de vulnerabilidades em aplicações em execução.

A integração do DAST nos pipelines de CI/CD promove segurança contínua, mas deve ser realizada de forma estratégica, combinando tuning das ferramentas, revisão dos resultados e automação equilibrada, de modo a maximizar a eficácia sem comprometer a velocidade de entrega de software.

6. Referências

Referências

[1] Lyubka Dencheva. Comparative analysis of static application security testing (sast) and dynamic application security testing (dast) by using open-source web application penetration testing tools. Msc thesis, National College of Ireland, Dublin, Ireland, 2022. URL `https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021`. Supervised by Mark Monaghan.

[2] OWASP Foundation. OWASP Top 10:2021 - A02:2021 – Cryptographic Failures. `https://owasp.org/Top10/A02_2021-Cryptographic_Failures/`, 2021.

[3] Agung Maulana Putra. 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)*, Bogor, Indonesia, 2022. IEEE. doi: 10.1109/ICOSNIKOM56551.2022.10034883.