Revisão Multifocal sobre ferramentas de teste e estratégias de segurança de APIs em microsserviços

Ian José Soares Mendes[1], **Lyanderson S. Rodrigues**[1], **Rubens A. da S. Sousa**[1]
Ismayle de S. Santos[1]

[1]Universidade Estadual do Ceará (UECE) – CCT - Ciência da Computação
Fortaleza – CE – Brazil

{ian.soares, lyanderson.rodrigues, abraao.sousa}@aluno.uece.br
ismayle.santos@uece.br

Abstract. *Communication in microservice architectures has become increasingly common, requiring attention to security testing to ensure the quality of APIs. This study, through a multifocal review, investigated security tools and strategies applied to API development. Fifty articles from white literature and 270 from gray literature were analyzed. The most cited tools were RESTest (6), OWASP ZAP (6) and Postman (4). Among the strategies, Spring Security (93), JWT (81) and API Gateway (37) stood out. The main tests involved input validation (20), vulnerability detection (19) and authentication/authorization (14), with a focus on mitigating threats such as SQL Injection (11), configuration flaws (10) and OWASP Top Ten (8).*

Resumo. *A comunicação em arquiteturas de microsserviços tem se tornado cada vez mais comum, exigindo atenção aos testes de segurança para garantir a qualidade das APIs. Este estudo, através de uma revisão multifocal, investigou ferramentas e estratégias de segurança aplicadas ao desenvolvimento de APIs. Foram analisados 50 artigos da literatura branca e 270 da cinza. As ferramentas mais citadas foram RESTest (6), OWASP ZAP (6) e Postman (4). Entre as estratégias, destacaram-se Spring Security (93), JWT (81) e API Gateway (37). Os principais testes envolvem validação de entrada (20), detecção de vulnerabilidades (19) e autenticação/autorização (14), com foco em mitigar ameaças como SQL Injection (11), falhas de configuração (10) e OWASP Top Ten (8).*

1. Introdução

Nos últimos anos, a utilização de microsserviços entre os desenvolvedores aumentou. Em 2021, 1.200 profissionais de TI foram entrevistados e os dados demonstraram que 78% utilizavam arquitetura de microsserviços em suas aplicações e ainda pretendiam expandir o uso dessa abordagem [IBM, 2021]. A utilização da arquitetura de microsserviços possibilita soluções que oferecem independência, escalabilidade e manutenibilidade [Domingos and Farina 2020].

Ao utilizar arquitetura de microsserviços, a comunicação é realizada por meio do uso de APIs (Application Programming Interface), que se tornou mais comum, dada a possibilidade de dividir a aplicação em componentes menores chamados serviços, que são independentes e trocam informações sensíveis entre si. Segundo relatório da Salt Security, 94% das empresas entrevistadas sofreram ataques de segurança em suas APIs

que estavam em produção em 2022 [Salt Security, 2023], sendo que 66% desses ataques se aproveitaram de vulnerabilidades registradas no OWASP API Security Top 10[1].

Dessa forma, a segurança torna-se importante para garantir a conformidade e a qualidade do software. Surge, assim, a viabilidade de testes de segurança voltados a APIs, visando identificar vulnerabilidades e falhas que possam comprometer a integridade, disponibilidade e confidencialidade dos sistemas (FELDERER et al., 2016). Além disso, o uso de ferramentas de teste e estratégias de segurança em APIs é importante para auxiliar o processo de testagem e desenvolvimento.

A adoção de medidas que possam garantir a segurança em APIs é necessária durante o processo de desenvolvimento de software. Investigar quais ferramentas de teste e quais estratégias implementar para promover a segurança em APIs torna-se, portanto, uma necessidade para auxiliar as equipes de teste e desenvolvimento na escolha da ferramenta e/ou estratégia a ser utilizada. Diante disso, o presente trabalho busca explorar o estado da arte, por meio de uma revisão multifocal da literatura [Garousi et al. 2016], sem limite temporal, buscando na literatura branca e cinza por ferramentas que realizem testes de segurança e por estratégias voltadas à segurança das APIs no contexto de microsserviços.

Para este trabalho, é adotada a seguinte divisão em seções: (i) Introdução, que apresenta o contexto atual sobre a segurança em APIs em microsserviços e o objetivo do estudo; (ii) Trabalhos relacionados, que apresenta estudos semelhantes que buscaram ferramentas e estratégias de segurança para APIs; (iii) Metodologia, na qual é descrito o processo adotado para a condução da revisão multifocal; (iv) Resultados, onde são apresentados os achados conforme a literatura, com as ferramentas e estratégias identificadas; (v) Discussões, na qual é feita uma análise sobre os resultados obtidos com base na literatura; e (vi) Conclusões, que retomam a síntese inicial e apresentam desfechos com considerações finais e sugestões para trabalhos futuros.

2. Trabalhos Relacionados

Esta seção apresenta trabalhos semelhantes que contribuíram na busca por ferramentas de teste e estratégias de segurança em APIs. São destacadas as abordagens adotadas, os contextos e limitações, bem como a contribuição que o presente estudo realiza, complementando os trabalhos já existentes.

Ghani et al. [Ghani et al. 2019], por meio de uma revisão sistemática da literatura, exploraram os desafios no ambiente de testes de microsserviços e as estratégias utilizadas para mitigá-los. Para isso, utilizaram parâmetros como segurança, desempenho, rastreabilidade, compatibilidade, complexidade, eficácia e escalabilidade, que asseguram a qualidade dessas aplicações. Entretanto, entre os estudos que apresentaram como contribuição alguma ferramenta, apenas um contemplou diretamente o aspecto de testes de segurança em APIs, não possibilitando discussões aprofundadas sobre a temática

Visando encontrar mecanismos para detecção, resposta e mitigação de ameaças em APIs, Diaz-Rojas et al. [Díaz-Rojas et al. 2021] conduziram um mapeamento sistemático com o objetivo de auxiliar o desenvolvimento seguro. Para a condução do estudo, os autores optaram pela categorização das técnicas identificadas de acordo com o tipo de ameaça. Observaram a possibilidade de utilizar os mecanismos encontrados para orientar

[1] `https://owasp.org/API-Security/editions/2023/en/0x00-header/`

um experimento empírico como forma estratégica de desenvolvimento para APIs. Dessa forma, o estudo não investigou ferramentas que pudessem auxiliar durante os testes de segurança.

De modo semelhante, Hannousse e Yahiouche [Hannousse and Yahiouche 2021] conduziram outro mapeamento sistemático voltado para estratégias de segurança destinadas à mitigação de ameaças conhecidas em arquiteturas de microsserviços, utilizando, na condução do mapeamento, a técnica de snowballing. Nesse sentido, foi realizada uma organização ontológica que possibilita identificar, conforme a ameaça, as estratégias e sugestões de como implementá-las na arquitetura de microsserviços. Embora os resultados se concentrem em frameworks, o estudo não menciona ferramentas que possam garantir que a adoção dessas estratégias de segurança em APIs mitigue as vulnerabilidades, uma vez que não há uma relação direta entre ferramenta, estratégia e ameaça.

Garcia et al. [García et al. 2023] realizaram outro mapeamento sistemático com o objetivo de identificar técnicas, métodos e estratégias aplicáveis durante a etapa de testes de APIs. Com esse propósito, os autores identificaram os tipos de testes executados, incluindo os de segurança, e realizaram uma síntese narrativa dos resultados coletados. Apesar disso, as estratégias presentes na literatura selecionada estavam, em sua maioria, voltadas para testes unitários e automatizados, deixando em aberto a necessidade de identificar mais estratégias que tratem especificamente da segurança.

De modo semelhante, Panahande e Miller [Panahande and Miller 2023] focaram, em sua revisão sistemática, na investigação dos testes em microsserviços. Para isso, realizaram comparações entre os métodos e abordagens utilizados na condução desses testes. Entretanto, neste estudo, a maioria dos artigos analisados considerava cenários com falhas desconhecidas, ou seja, partia-se do pressuposto de que os microsserviços não eram resilientes. Essa limitação reforça a necessidade de pesquisas que se proponham a analisar ferramentas e estratégias de segurança capazes de auxiliar na resposta a ameaças em APIs.

Tabela 1. Comparação dos trabalhos relacionados

	Lit. branca	Lit. cinza	Foco em API	Ferramentas	Estratégias
Ghani et al.	✓	✗	✓	✓	✗
Diaz-Rojas et al.	✓	✗	✓	✓	✗
Hannousse and Yahiouche	✓	✗	✓	✗	✗
Garcia et al.	✓	✗	✓	✗	✓
Panahandeh and Miller	✓	✗	✗	✓	✓
Trabalho proposto	✓	✓	✓	✓	✓

Desta forma, este estudo visa atuar em duas vertentes em paralelo, ou seja, em ferramentas para testes de segurança e estratégias para implementação de segurança em APIs de microsserviços.

3. Método

Para a condução deste estudo, optou-se pela realização de uma revisão multifocal, que tem como objetivo considerar tanto a literatura branca quanto a cinza, proporcionando uma visão mais abrangente das fontes e dos dados disponíveis [Garousi et al. 2019]. Considerando o método adotado para a execução da pesquisa, foi realizada a análise de ambas as

literaturas em busca de ferramentas de teste e estratégias de segurança aplicáveis a APIs no contexto de microsserviços.

3.1. Questões de Pesquisa

Este estudo teve como objetivo identificar as ferramentas utilizadas para a realização de testes de segurança em APIs de microsserviços, bem como as estratégias de segurança aplicáveis a esse contexto. Além disso, buscou-se levantar os tipos de teste de segurança suportados e a cobertura de vulnerabilidades em APIs. Com base nisso, foram elaboradas as seguintes questões de pesquisa:

- **RQ1: Quais as ferramentas usadas para testar a segurança das APIs?**
  Considerando que os sistemas de microsserviços fazem uso de diferentes APIs, realizar os testes de segurança exige o apoio de ferramentas. Assim, ter uma visão geral das opções pode facilitar a escolha das mais adequadas ao escopo do teste.
- **RQ2: Quais as estratégias de segurança usadas para o desenvolvimento das APIs?**
  Levando em conta que diferentes APIs são utilizadas por sistemas baseados em microsserviços, o desenvolvimento dessas APIs requer a adoção de estratégias voltadas para a segurança. Assim, ter uma visão geral das estratégias disponíveis, as quais podem envolver tecnologias, protocolos ou soluções, pode facilitar a escolha daquelas mais adequadas ao desenvolvimento da API.
- **RQ3: Quais são os testes de segurança suportados pelas ferramentas?**
  Considerando que o profissional de segurança possui conhecimento sobre a arquitetura, cabe a ele realizar o planejamento e a execução dos testes de segurança em APIs. As ferramentas, devido ao seu escopo, possibilitam a testagem de determinados tipos de vulnerabilidades.
- **RQ4: Qual a cobertura de vulnerabilidades de API fornecida pelas soluções?**
  Estar ciente do alcance das ferramentas permite que os profissionais realizem testes mais assertivos na identificação de vulnerabilidades em um sistema, considerando que essa é uma das principais atividades das equipes de segurança.

3.2. Protocolo

Nesta subseção, apresenta-se o protocolo adotado para a revisão sistemática multifocal sobre ferramentas de testes e estratégias de segurança para APIs de microsserviços. O protocolo contempla: critérios de inclusão e exclusão, que orientam a seleção dos estudos pertinentes; strings de busca, para garantir a abrangência na coleta das publicações relevantes; seleção dos estudos, que descreve as etapas adotadas para a análise; snowballing, que realiza a identificação dos trabalhos por meio das referências dos estudos incluídos; e, por fim, a extração de dados, indicando quais foram os dados coletados para responder às perguntas de pesquisa formuladas.

Seleção das bases de dados

Foram selecionadas quatro bases de relevância para a área, escolhidas de acordo com a disponibilidade de acesso via instituição de ensino, sendo elas: Scopus[2], ISI Web of Science[3], IEEE Digital Library[4] e El Compendex[5]. Para a literatura cinza, foram

[2] `http://www.scopus.com/`
[3] `http://www.isiknowledge.com/`
[4] `http://ieeexplore.ieee.org/`
[5] `http://www.engineeringvillage.com/`

selecionadas três bases, visando ampliar ao máximo a variedade de resultados obtidos, sendo elas: Google[6], Stack Exchange[7] e GitHub[8]. Na base de dados do Google, realizaram a análise das 10 primeiras páginas por ordem de relevância, semelhante ao trabalho de Bakhtin et al. [Bakhtin et al. 2022].

Critérios de inclusão e exclusão

Para a seleção dos estudos, foi realizado o estabelecimento dos critérios de inclusão, sendo considerados artigos escritos em inglês, artigos que relatem sobre ferramentas de testes de segurança em APIs de microsserviços ou estratégias que implementem segurança em APIs de microsserviços. Foram excluídos artigos que sejam estudos secundários, que não respondam às questões de pesquisa e que não estejam disponíveis. Além disso, o período de publicação não foi considerado pelos critérios.

String de busca

A pesquisa utilizou duas strings de busca distintas: uma para a literatura branca e outra para a literatura cinza, com alterações entre elas. Bases como Stack Exchange e GitHub apresentaram limitações quanto ao número de palavras-chave e ao não reconhecimento de operadores booleanos além do AND. A construção das strings seguiu o guia de Kitchenham et al. [Kitchenham et al., 2007], garantindo rigor metodológico. Ambas as strings foram elaboradas com base no critério PICOC (Population, Intervention, Context, Outcome, Comparison), assegurando alinhamento com os objetivos da revisão, conforme mostra a Tabela 2.

Tabela 2. Termos do PICOC e correspondentes na string de busca

Termos do PICOC	**Termos da string de busca**
População	(“API*” OR “Application Programming Interface” OR “Restful”)
Intervenção	(“Security” OR “secure”)
Comparação	-
Resultado	(“Tool*” OR “Framework*” OR “Technolo*” OR “Platform*”)
Contexto	(“microservice*” OR “micro-service*” OR “micro service*”)

Para otimizar a busca, foram empregados os operadores booleanos “AND” e “OR”, além do caractere especial “*”, que permite captar variações no singular e plural das palavras-chave. Essa abordagem buscou maximizar a abrangência e a relevância dos estudos selecionados, como mostra a Tabela 3.

[6]http://google.com/
[7]http://stackexchange.com/
[8]http://github.com/

Tabela 3. Strings de busca utilizadas para a pesquisa na literatura branca e cinza

Fonte	String de busca
Literatura branca	(microservice* OR micro-service* OR “micro service*”) AND (“API*” OR “Application Programming Interface” OR Restful) AND (“Security” OR “secure”) AND (“Tool*” OR “Framework*” OR “Technolo*” OR “Platform*”)
Literatura cinza	(”microservice”AND ”API”AND ”security”)

Seleção dos estudos

Finalizada a consolidação da string de busca utilizada na literatura branca e cinza, realizaram a coleta dos estudos, exportando em formato .bib e adicionando na ferramenta Parsifal, que realiza o gerenciamento de revisões sistemáticas. Após a adição na ferramenta, deram início ao processo de triagem conforme os critérios de inclusão e exclusão definidos, o qual consistiu em: 1°) remoção de duplicatas, 2°) leitura do abstract, título e palavras-chave, 3°) leitura por completo, como mostra a Figura 1.

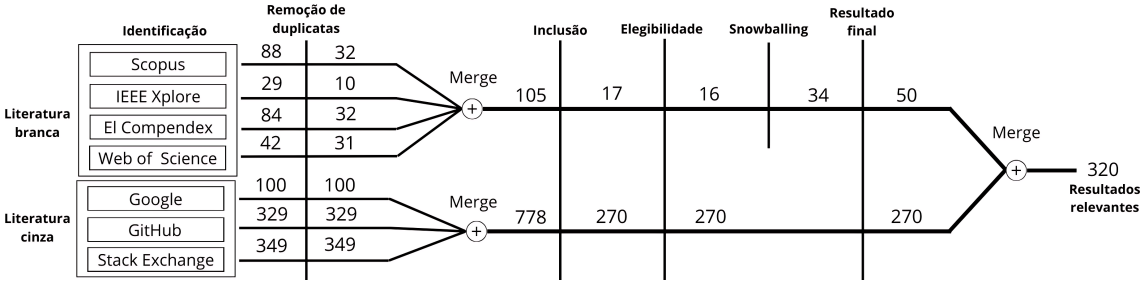

Figura 1. Processo de seleção dos estudos

Ao coletarem os trabalhos, as bases de dados da literatura branca reuniram os seguintes quantitativos: Scopus (88), IEEE (29), Web of Science (42), El Compendex (84), totalizando 243 trabalhos. Desses, foram removidos 56 estudos da Scopus, 19 da IEEE, 52 da El Compendex e 11 da Web of Science, totalizando 138 trabalhos duplicados eliminados. Assim, 105 trabalhos passaram para a etapa de inclusão, na qual foi realizada a leitura do título, abstract e palavras-chave, resultando na inclusão de 17 artigos na etapa de elegibilidade, que consistiu na leitura completa do trabalho. O resultado final dessa análise foram 16 trabalhos relevantes que passaram para a extração de dados e snowballing da literatura branca.

Em relação à literatura cinza, as bases Google (100), GitHub (329) e Stack Exchange (349) totalizaram 778 achados, sem itens duplicados, elegendo 270 para a etapa de inclusão e permanecendo como resultados relevantes para o trabalho. Assim, realizaram a junção dos resultados à literatura branca, totalizando 320 resultados relevantes para a pesquisa.

Avaliação de qualidade dos resultados

Para avaliar a qualidade dos estudos, construíram um formulário de avaliação, disponível no repositório, com respostas classificadas como Sim = 1 e Não = 0. Ao final da avaliação, os melhores trabalhos receberiam 7 pontos, demonstrando que o artigo atendia ao objetivo das perguntas. Após a avaliação, seguiram a estratégia de [ul Haq 2008],

adotando uma abordagem que prioriza a seleção de todos os artigos, sem impor um limite arbitrário que pudesse excluir estudos relevantes.

Formulário de extração de dados

Para responder às perguntas de pesquisa, construíram um formulário de extração de dados, disponível no repositório. O formulário inclui campos bibliográficos, como autores e conferências/revistas. Os outros campos levantavam e buscavam identificar quais são as ferramentas de teste de segurança e as estratégias de segurança mencionadas nos trabalhos, como nome, tipos de testes de segurança suportados, cobertura de vulnerabilidades, entre outros campos.

4. Resultados

Nesta seção, apresentam os resultados do processo de pesquisa, após a avaliação dos 50 estudos da literatura branca e 270 resultados da literatura cinza, totalizando 320 resultados avaliados com o objetivo de responder às questões de pesquisa que foram anteriormente estabelecidas.

4.1. RQ1: Quais as ferramentas usadas para testar a segurança das APIs?

Após analisarem os dados, foi possível notar um número diverso de ferramentas utilizadas para testar a segurança das APIs em sistemas de microsserviços. A Figura 2 apresenta o gráfico das principais ferramentas analisadas na literatura branca e cinza para os testes das APIs.

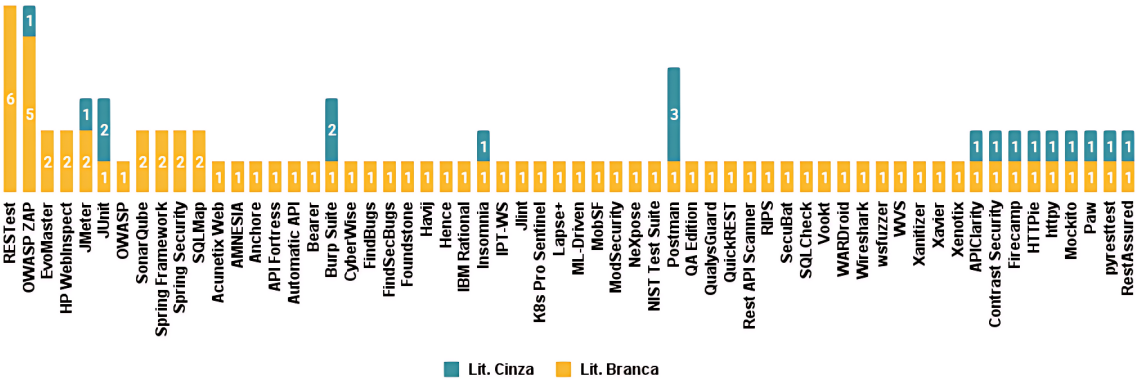

Figura 2. Ferramentas de testes de segurança em APIs

De acordo com a Figura 2, que relata os resultados, as ferramentas RESTest (6) e OWASP ZAP (6) são as principais utilizadas para testar a segurança das APIs. Logo em seguida, aparecem o Postman (4), JUnit (3), JMeter (3), EvoMaster (2), de uso gratuito, e Burp Suite (3), sendo esta a única ferramenta citada que possui uma versão paga.

4.2. RQ2: Quais as estratégias de segurança usadas para o desenvolvimento das APIs?

Na Figura 2, apresentam as estratégias voltadas para segurança descobertas e que, consequentemente, realizam a proteção das APIs dos microsserviços.

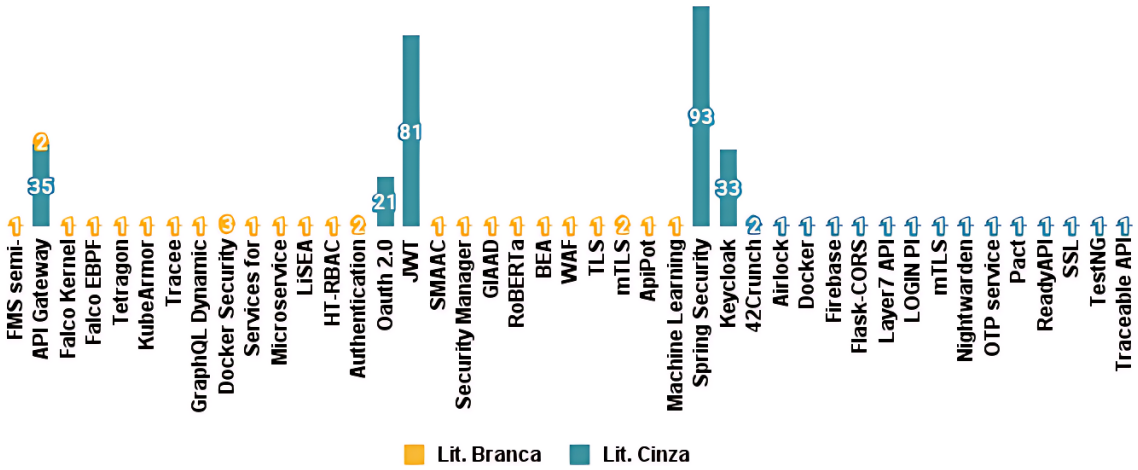

Figura 3. Estratégias de segurança voltadas para APIs

Nota-se que, na Figura 3, na literatura cinza, o Spring Security (93), juntamente com o JWT (81) — respectivamente, um framework de segurança e uma estratégia de autorização — foram os mais utilizados. Outras três estratégias também se destacam pelo elevado número de utilizações, sendo elas: API Gateway (37), Keycloak (33) e OAuth 2.0 (21).

4.3. RQ3: Quais os testes de segurança suportados pelas ferramentas?

Tendo em vista a existência de uma variedade de estratégias e testes com abordagens e intuitos diferentes, buscaram selecionar as categorias mais presentes na literatura. Na Figura 4, estão apresentados os testes encontrados, sendo eles de diferentes tipos e objetivos, variando entre criptografia, validações e scanners até a reprodução de situações reais de uma invasão.

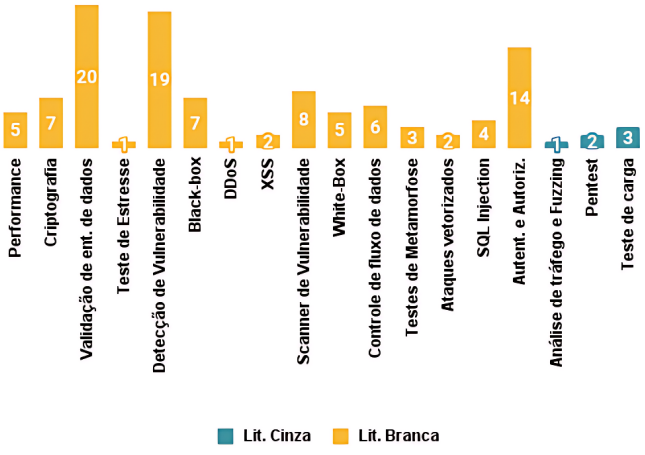

Figura 4. Tipos de testes de segurança suportados pelas ferramentas

Segundo a Figura 4, existe uma maior tendência para os testes que se propõem à validação da entrada de dados (20), à detecção de vulnerabilidades (19) e à segurança dos processos de autenticação e autorização (14). Por outro lado, o menor número na análise

está relacionado aos testes que simulam ataques ou exigem mais das APIs quanto à sua infraestrutura, como performance (5), DDoS (1) e análise de tráfego e fuzzing (1).

4.4. RQ4: Qual a cobertura de vulnerabilidades de API fornecida pelas soluções?

Para entenderem as limitações das ferramentas e estratégias que auxiliam no processo de testes de segurança, investigaram qual o alcance de vulnerabilidades que cada uma delas apresenta. A Figura 5 apresenta os níveis distintos de suporte, conforme encontrado.

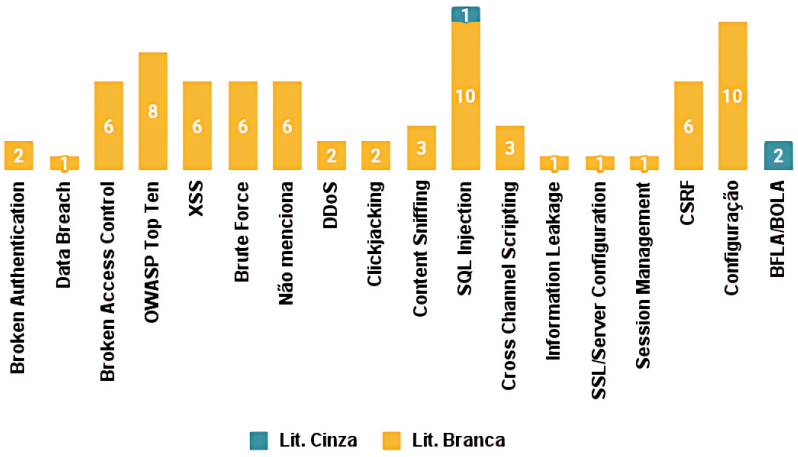

Figura 5. Cobertura de vulnerabilidades fornecida pelas ferramentas de teste de segurança de APIs

Em relação à Figura 5, observam que a vulnerabilidade mais tratada pelas ferramentas e estratégias é a injeção de código SQL (11), possivelmente devido à característica das APIs de receberem entrada de dados, o que permite maior exploração desse tipo de ataque por invasores. Além disso, em referência ao OWASP Top Ten (8), buscaram mitigar as vulnerabilidades apresentadas pelo relatório, que agrupa os riscos mais comuns. As vulnerabilidades de configuração (10) também estiveram presentes, embora em número menor.

5. Discussões

Após a apresentação dos resultados para cada questão de pesquisa, esta seção tem como objetivo discutir as principais descobertas e desafios de acordo com esse conjunto de dados. Ademais, apresentam algumas limitações que podem ser consideradas como ameaças à validade da pesquisa.

De início, os resultados referentes à RQ1, sobre quais são as ferramentas utilizadas para testar a segurança das APIs, retornaram dados relevantes que apontam uma predominância na utilização de ferramentas gratuitas, como OWASP ZAP, RESTest e Burp Suite, que possuem funcionalidades semelhantes para a realização de testes de segurança em APIs. No entanto, os resultados indicam uma menor adoção da ferramenta Burp Suite, o que pode ser explicado pela existência de uma versão paga, enquanto suas concorrentes são totalmente gratuitas.

Além disso, os resultados relacionados à RQ2, sobre as estratégias de segurança utilizadas no desenvolvimento das APIs, mostram uma padronização na adoção dessas

práticas, já que as cinco principais concentram mais de 86% das utilizações, voltadas principalmente para autorização e autenticação, com destaque para o API Gateway. As estratégias citadas são gratuitas, o que contribui para sua popularidade. Ressalta-se, no entanto, que a maioria dos resultados foi encontrada na literatura cinza, abrindo espaço para novos estudos sobre sua aplicação em contextos de microsserviços.

Adicionalmente, o estudo buscou responder à RQ3, sobre quais são os testes de segurança suportados pelas ferramentas. Os resultados mostram que a validação da entrada de dados, a detecção de vulnerabilidades e a segurança dos processos de autenticação e autorização são os principais aspectos testados, com mais de 48% das utilizações abordando um desses três testes. No entanto, testes que simulam ataques reais, como testes de estresse, DDoS, análise de tráfego e fuzzing, ainda são pouco explorados. Os resultados se concentram na literatura cinza, evidenciando uma lacuna na literatura formal, que pouco trata dos testes de segurança suportados por ferramentas e estratégias voltadas à segurança de APIs.

Por fim, para responder à RQ4, sobre a cobertura de vulnerabilidades fornecida pelas ferramentas de teste de segurança de APIs, os resultados demonstraram que a vulnerabilidade mais explorada é a injeção de código SQL, por afetar diretamente uma parte essencial do funcionamento de uma API: a entrada de dados. Outras vulnerabilidades cobertas pelas funcionalidades das ferramentas e estratégias são as listadas no OWASP Top 10, que reúne as principais falhas em APIs. No entanto, falhas como Broken Authentication, Session Management e DDoS foram pouco abordadas, abrindo espaço para estudos futuros que investiguem a aplicação dessas ferramentas em contextos de microsserviços voltados à testagem dessas vulnerabilidades.

Mesmo que o trabalho tenha contemplado uma quantidade significativa de ferramentas de segurança e estratégias para proteção das APIs, destacaram-se as seguintes ameaças à validade: i) possíveis discordâncias quanto aos estudos aprovados durante o processo de seleção; ii) ausência de filtros que removessem anúncios nas 10 primeiras páginas do Google, fonte da literatura cinza. Para minimizar essas ameaças, adotaram as seguintes ações: i) aplicaram revisão por pares e, em caso de discordância, contaram com um terceiro revisor para garantir julgamento imparcial; ii) excluíram sites desalinhados ao objetivo da pesquisa, sem considerar links adicionais às páginas especificadas, evitando comprometer a qualidade das evidências.

6. Conclusão

Após a revisão multifocal, alcançaram 320 resultados. Para isso, consultaram fontes da literatura branca e cinza, o que permitiu uma quantidade robusta e variada de análises. Em seguida, aplicaram uma metodologia com diferentes filtros, buscando um tratamento rigoroso das extrações para obter resultados que respondessem às questões de pesquisa.

Contudo, as informações presentes neste estudo fornecem opções de ferramentas que auxiliam nos testes de segurança, bem como suas limitações, além de estratégias aplicáveis ao desenvolvimento de APIs. Para trabalhos futuros, sugerem experimentos empíricos que confrontem ferramentas e estratégias, a fim de analisar sua efetividade na mitigação ou identificação de vulnerabilidades, servindo como suporte a pesquisadores e profissionais da área de segurança da informação.

Referências

Bakhtin, A., Maruf, A. A., Cerny, T., and Taibi, D. (2022). Survey on tools and techniques detecting microservice api patterns.

Domingos, L. and Farina, R. (2020). MicroserviÇos: um estudo de caso apontando suas potencialidades. *Revista Interface Tecnológica*, 17:18–30.

Díaz-Rojas, J. A., Ocharán-Hernández, J. O., Pérez-Arriaga, J. C., and Limón, X. (2021). Web api security vulnerabilities and mitigation mechanisms: A systematic mapping study. In *2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pages 207–218.

García, J. C., Hernández, J. O. O., Arriaga, J. C. P., and Riaño, H. J. L. (2023). Advances in web api testing: A systematic mapping study. In *2023 Mexican International Conference on Computer Science (ENC)*, pages 1–8.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2016). The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, EASE '16, New York, NY, USA. Association for Computing Machinery.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, 106:101–121.

Ghani, I., Wan-Kadir, W. M., Mustafa, A., and Babir, M. I. (2019). Microservice testing approaches: A systematic literature review. *International Journal of Integrated Engineering*, 11(8):65–80.

Hannousse, A. and Yahiouche, S. (2021). Securing microservices and microservice architectures: A systematic mapping study. *Computer Science Review*, 41:100415.

Panahande, M. and Miller, J. (2023). A systematic review on microservice testing. *Research Square*.

ul Haq, S. M. (2008). Assessing the quality of primary studies in systematic reviews. *International Journal of Health Promotion and Education*, 46:139–141.