Um mapeamento sistemático de aplicações de código aberto para busca de falhas de segurança no processo de desenvolvimento de software

Marcelo Martins Pinto[1]

[1]Instituto Metrópole Digital (IMD)
Universidade Federal do Rio Grande do Norte (UFRN)
CEP: 59078-900 – Natal – RN – Brazil

pintomarc@gmail.com

Abstract. *When adopting DevOps practices, the security aspect cannot be ignored. Therefore, the software development life cycle must include a security check on the source code of applications. This work aimed to map models, techniques and open source applications that have the potential to be used as a basis for the development of a centralized management project for applications and their associated infrastructures. It showed that there are several applications that can be used for this purpose and identified the need for further in-depth analysis and other studies to evaluate the recommendations and best practices for implementing software for security purposes.*

Resumo. *Na adoção das práticas de DevOps o aspecto de segurança não pode ser deixado de lado. Assim, o ciclo de vida do desenvolvimento de software precisa incluir uma verificação de segurança no código-fonte das aplicações. Este trabalho objetivou mapear modelos, técnicas e aplicações de código aberto que têm potencial para uso como base no desenvolvimento de projeto de gerenciamento centralizado de aplicações e suas infraestruturas agregadas. Mostrou que há diversas aplicações que podem ser utilizadas com essa finalidade e identificou a necessidade de maior aprofundamento e outros estudos para avaliar as recomendações e melhores práticas para implementação de softwares com a finalidade de segurança.*

1. Introdução

Apesar de se ouvir falar bastante em DevOps[1], que poderia ser designada como um conjunto de práticas e valores culturais, que surgiu quando a indústria de desenvolvimento de software percebeu que os silos existentes entre as equipes de desenvolvimento e infraestrutura estavam sendo prejudiciais e que a adoção da prática ajudaria a melhorar a capacidade, eficiência, eficácia, performance e produtividade, além da redução de custos, sua adoção prática não é tão trivial, conforme citado por [Luz et al., 2019, Khan et al., 2022], por ser necessário um trabalho de mudança de paradigmas. Além da necessidade de quebra dos silos já existentes entre desenvolvimento e infraestrutura, há necessidade cada vez mais latente de se incluir o elemento segurança, tão em voga atualmente, tornando obrigatório que os softwares desenvolvidos sejam cada vez mais seguros, tornando a

[1]DevOps - Development and Operations

preocupação compartilhada agora entre as equipes de desenvolvimento, infraestrutura e segurança tanto das aplicações desenvolvidas quanto a infraestrutura que dá suporte a elas, DevSecOps[2].

1.1. Motivação

O contexto das vulnerabilidades de segurança, cada dia mais exploradas pelos cibercriminosos, que, através da exploração dessas falhas, conseguem penetrar em um sistema, ou na própria rede das empresas, sequestram ou se apropriam de dados sensíveis e relevantes, demonstra a necessidade de se incorporar a segurança no ciclo de vida do desenvolvimento do software. Dessa maneira, visando desenvolver uma solução integrada, que possibilite o mapeamento das configurações, do ciclo de vida das aplicações e os recursos utilizados por elas, no contexto de DevSecOps, alinhado à existência de inúmeras soluções pagas e de código aberto para análise de vulnerabilidades de segurança em código-fonte, o presente trabalho se apresenta com a proposta de prospecção das soluções de código aberto disponíveis para a finalidade apontada.

1.2. Objetivos

A pesquisa realizada é baseada nos objetivos geral e específicos, que são apresentados a seguir.

1.2.1. Objetivo Geral

O presente trabalho visa identificar e mapear aplicações, modelos, técnicas ou outras soluções, de código fonte aberto, que tenham potencial para uso como parte integrante do desenvolvimento de uma solução para controle e gerenciamento centralizado de aplicações e serviços, no contexto de DevSecOps, para análise de vulnerabilidades no processo de desenvolvimento de software.

1.2.2. Objetivos Específicos

- Objetivo 1: Listar as recomendações mais frequentes de uso e adoção de ferramentas para análise de vulnerabilidade;
- Objetivo 2: Listar técnicas, modelos, aplicações e ferramentas de código fonte aberto, passíveis de serem utilizadas no desenvolvimento de solução para DevSecOps;
- Objetivo 3: Identificar linguagens de programação mais utilizadas para análise de vulnerabilidade de segurança.

1.3. Resumo dos principais resultados

Considerando o que foi possível concluir com o trabalho, é apresentado um quadro comparativo entre as principais soluções encontradas, não havendo uma recomendação específica sobre qual ferramenta é ideal para análise de vulnerabilidade, uma vez que todas têm suas características específicas, positivas e negativas, conforme o contexto. Ademais, destacamos a adoção do Python como a linguagem de programação mais citada e o

[2]DevSecOps - Development, Securirty and Operations

uso da automação de pipelines CI/CD através de CodeQL e uso do *Github Actions* para incorporação ao ciclo de vida do desenvolvimento das aplicações, como mais citados no que concerne à análise de vulnerabilidades.

1.4. Estrutura do trabalho

As seções deste trabalho estão organizadas da seguinte forma:

- **Seção 1** - Introduz o problema, apresenta a motivação e os objetivos do trabalho;
- **Seção 2** - Apresenta a metodologia utilizada relacionada à pesquisa;
- **Seção 3** - Demonstra os resultados obtidos;
- **Seção 4** - Considerações finais, breve apresentação dos resultados obtidos, limitações e dificuldades encontradas e trabalhos futuros.

2. Metodologia

Nesta seção, faremos a apresentação da metodologia utilizada para realizar a pesquisa.

2.1. Objetivo do estudo

O presente estudo visa mapear aplicações de código-fonte aberto, bem como rotinas e padrões que possam ser utilizados em ciclos de vida de DevSecOps, com o objetivo de fazer análise de vulnerabilidade de código-fonte no processo de desenvolvimento de software.

2.2. Questões de pesquisa

- RQ1 O que a literatura fala acerca das recomendações sobre análise de vulnerabilidades no contexto de DevOps?
- RQ2 Há aplicações de licenciamento público que fazem pesquisa de vulnerabilidade em código-fonte nos repositórios, como github, ou gitlab?
- RQ3 Quais aplicações, modelos ou rotinas poderiam ser utilizadas como base para pesquisas automatizadas de vulnerabilidades de segurança em códigos-fonte, no processo de desenvolvimento de software?
  - RQ3.1 São aplicações de código público que podem ser incorporadas a um software de terceiro?
  - RQ3.2 Que tipos de varredura de falhas de segurança ela busca? Por exemplo: *sql injection*, privacidade de dados pessoais, *bug* no código, CVEs etc
  - RQ3.4 Qual base de dados de segurança ela utiliza?
  - RQ3.5 Qual a frequência de atualização dessa base de dados?
  - RQ3.6 Qual o tempo desde a última atualização da aplicação, após a publicação inicial?
  - RQ3.7 A aplicação é utilizada/referenciada por outros autores/desenvolvedores?
  - RQ3.8 Qual a linguagem de desenvolvimento dessa aplicação? Java, Python, PHP, Javascript etc

  Intervenção: Avaliação do potencial de uso das regras e modelos indicados nos artigos, padrões ou aplicações desenvolvidas como parte integrante de projeto de *framework* para controle centralizado de aplicações e serviços.

Controle: Revisões sistemáticas, artigos, teses e dissertações sobre aplicações de segurança em código-fonte obtidas na Internet, bem como aplicações e projetos de código aberto encontrados no github.
População: Documentos publicados e projetos de soluções de código público disponíveis na internet para busca de vulnerabilidades de segurança em código-fonte de softwares.
Resultados: Visão geral sobre os métodos, técnicas, modelos e aplicações para validação de segurança de código-fonte e possível seleção de *software* como base para utilização em *framework* de controle centralizado de aplicações e serviços.
Aplicação: no contexto de DevOps.

2.3. *String* de busca e bases de dados

As fontes escolhidas para a pesquisa foram as disponíveis através da internet, no período da realização do trabalho, dando-se preferência às bases de dados científicas, bem como trabalhos disponibilizados em outros meios técnicos, como o Github e Google Scholar, por serem fontes de informações da área de TI passíveis de utilização para Revisão Sistemática de Literatura Cinzenta.

Foram considerados livros, capítulos de livros, revisões sistemáticas, artigos, teses e dissertações conduzidos por profissionais ou estudantes da área de tecnologia da informação, bem como implementações de modelos e aplicações de código aberto e tutoriais públicos disponibilizados no GitHub sobre o tema, por profissionais de TI e outros interessados na área, em língua inglesa ou portuguesa.

A Tabela 1 abaixo lista os argumentos de pesquisa utilizados em cada fonte pesquisada.

Fonte	Chave de busca	Outros critérios
GitHub	**1ª pesquisa**: code scanning AND vulnerability code scan; **2ª pesquisa**: vulnerability code scan; **3ª pesquisa**: "security code scanning tools" OR "code scanning"OR "vulnerability code scanning"OR devops	
Scopus/	**Pesquisa**: ("devops"OR "devsecops") AND ("framework"OR "application"OR "tool"OR "vulnerability code scanning"OR "cve")	
Google Scholar	**Pesquisa**:("open source security code scanning tools"OR "code scanning"OR "vulnerability code scanning") AND ("devops"OR "devsecops") AND "cve"	Ano de publicação de 2018 a 2023
ArXiv.org	**Pesquisa**: devsecops OR devops AND framework OR application OR tool OR "code scanning"OR vulnerability OR cve	
IEEE Xplore	**Pesquisa**: ("devops"OR "devsecops") AND ("framework"OR "application"OR "tool"OR "vulnerability code scanning"OR "cve")	

Tabela 1. Bases e argumentos de busca

2.4. Critérios de inclusão e exclusão

Para a seleção dos trabalhos, foram considerados os seguintes critérios, conforme relacionados:

2.4.1. Critérios de inclusão

IC1 Software, modelo, métodos e técnicas relacionadas ao tema, de uso público;
IC2 Documentação relevante citado por outro material relacionados ao tema [IC1];
IC3 Documentação relevante para o tema.

2.4.2. Critérios de exclusão

EC1 Material relacionado a plataforma proprietária;
EC2 Material publicado como artigo muito curto (menor que 4 páginas);
EC3 Material obsoleto publicado anteriormente a 2018 (5 anos);
EC4 Material não escrito em língua inglesa ou portuguesa;
EC5 Material não relacionado ao temo da pesquisa.

2.5. Critérios de avaliação de qualidade dos materiais selecionados

Como critérios de avaliação da qualidade do material utilizado, consideramos aqueles trabalhos publicados em periódicos, livros ou revistas ou apresentados em eventos, ou aprovados por banca examinadora quando se tratarem de trabalhos de conclusão de curso, mestrado ou doutorado, segundo os critérios indicados em [Garousi et al., 2019].

No que diz respeito às aplicações, modelos e técnicas, foram selecionadas as que possuem relacionamento ao tema de maneira indistinta, bem como foram atribuídos pesos para as divisões (*forks*), as observações (*watching*) e estrelas (*stars*) conforme indicados pelos usuários e visitantes das páginas dos projetos, registrados pelo Github, que foram utilizados como parte da fórmula utilizada para gerar a pontuação de cada projeto.

2.6. Procedimentos para extração de dados

O processo de extração de dados foi realizado de duas formas distintas, considerando as fontes de dados utilizadas.

2.6.1. Base de dados do Github

1. **Criação de planilhas** Para preparação da extração de dados foram montadas três planilhas no Google Planilhas contendo os dados obtidos através da pesquisa realizada na página do Github, conforme indicado na Tabela 1, denominadas 1ªPesquisa, 2ªPesquisa e 3ªPesquisa, contendo as seguintes colunas:
    - SEQ - Indica um número sequencial atribuído ao item na planilha;
    - NOME - Indica o nome do projeto no Github;
    - URL - Indica o endereço da página do projeto;
    - LINGUAGEM - Indica a linguagem de programação utilizada no projeto;

- *FORKS* - Indica a quantidade de divisões que o projeto sofreu, computadas a criação de *forks* realizadas no Github;
- *WATCHING* - Indica a quantidade de observadores do projeto, computadas quando algum usuário do Github marca o projeto como *Watching*;
- ESTRELAS - Indica a quantidade de marcações/curtidas realizadas através das buscas no Github;
- LICENÇA - Indica o tipo de licenciamento do conteúdo do projeto;
- PUBLICAÇÃO - Indica o ano da publicação do conteúdo do projeto;
- ALTERAÇÃO - Indica o ano da última alteração do conteúdo, sendo que para as alterações ocorridas no ano corrente (2023), foram indicados mês e ano;
- BANCO DE DADOS - Indica o banco de dados de pesquisa de vulnerabilidade indicado pelo autor do projeto;
- RESUMO - Contém um resumo do projeto, elaborado através da leitura do ***Readme*** ou outros elementos existentes no projeto.

2. **Coleta dos dados** O processo de coleta dos dados se deu de forma manual, copiando e colando as informações obtidas nas pesquisas nas planilhas correspondente, exceto o campo resumo, que foi preenchido posteriormente, após a leitura das informações disponíveis arquivo *Readme* do projeto ou outros arquivos disponíveis.
3. **Limpeza de dados duplicados**
   Com a conclusão da cópia e mapeamento das aplicações, procedemos junção de todas os projetos listados em uma única planilha com dados compilados), aplicando-se a ela uma primeira limpeza de dados, com a remoção dos projetos duplicados, bem como o acréscimo das seguintes colunas e classificação:
   - **É DO ASSUNTO?** - Campo booleano[3] de opção, para indicar se o projeto atende ao Critério de inclusão IC1, podendo escolher entre SIM—NÃO;
   - **É VAZIO?** - Campo booleano, calculado automaticamente, através de uma sequência padrão de caracteres (Regex) aplicada na coluna **resumo**, podendo ter os valores SIM—NÃO;
   - **É PROPRIETÁRIO?** - Campo booleano, calculado automaticamente, através de uma Regex aplicada na coluna **linguagem**, podendo ter os valores SIM—NÃO;
   - **PONTUAÇÃO?** - Campo numérico inteiro calculado, aplicando-se a seguinte regra, Se a coluna I (ano da publicação) for vazia ou igual a "-", o valor da pontuação será zero; Se não, calcula a quantidade de *forks* por ano, multiplicado pelo peso (0,5), somado com a quantidade de *Watching* por ano, multiplicado pelo peso (0,3), somando-se com a quantidade de estrelas por ano, multiplicado pelo peso (0,2), que é representada pela fórmula aplicada na planilha:

$$\begin{aligned} &SE(I2 ='' -''; 0; ARREDONDAR.PARA.CIMA(\\ &\quad E2/(SE(2023 - I2 = 0; 1; (2023 - I2))) * 0,5+ \\ &\quad F2/(SE(2023 - I2 = 0; 1; (2023 - I2))) * 0,3+ \\ &\quad G2/(SE(2023 - I2 = 0; 1; (2023 - I2))) * 0,2)) \end{aligned} \tag{1}$$

[3]Campo que só permite dois valores possíveis: Verdadeiro ou Falso

4. **Classificação e finalização do mapeamento** Feita a aplicação das fórmulas/regras indicadas acima e a releitura do campo Resumo para pre-enchimento dos campos de classificação adicionados, procedemos a criação de um *autofiltro* na planilha e finalizamos o processo de seleção das aplicações com a aplicação dos seguintes filtros nas colunas indicadas, conforme Tabela 2.

Tabela 2. Filtros de Dados Compilados

COLUNA	FILTRO APLICADO
É ASSUNTO?	SIM
É VAZIO?	NÃO
É PROPRIETÁRIO?	NÃO e N/A
PONTUAÇÃO?	VALOR $\geq$ 26

Destacamos que o filtro da pontuação foi selecionado como sendo o valor inteiro do desvio padrão calculado, conforme indicado na Figura 5.

.

3. Resultados

Os resultados apresentados foram apurados através da análise da planilha indicada onde foram levantados um total de 384 registros em 3 (três) pesquisas realizadas, tendo sido removidos 141 (cento e quarenta e um) registros por duplicação, restando um total de 243 (duzentos e quarenta e três) registros para serem , conforme Figura 1:

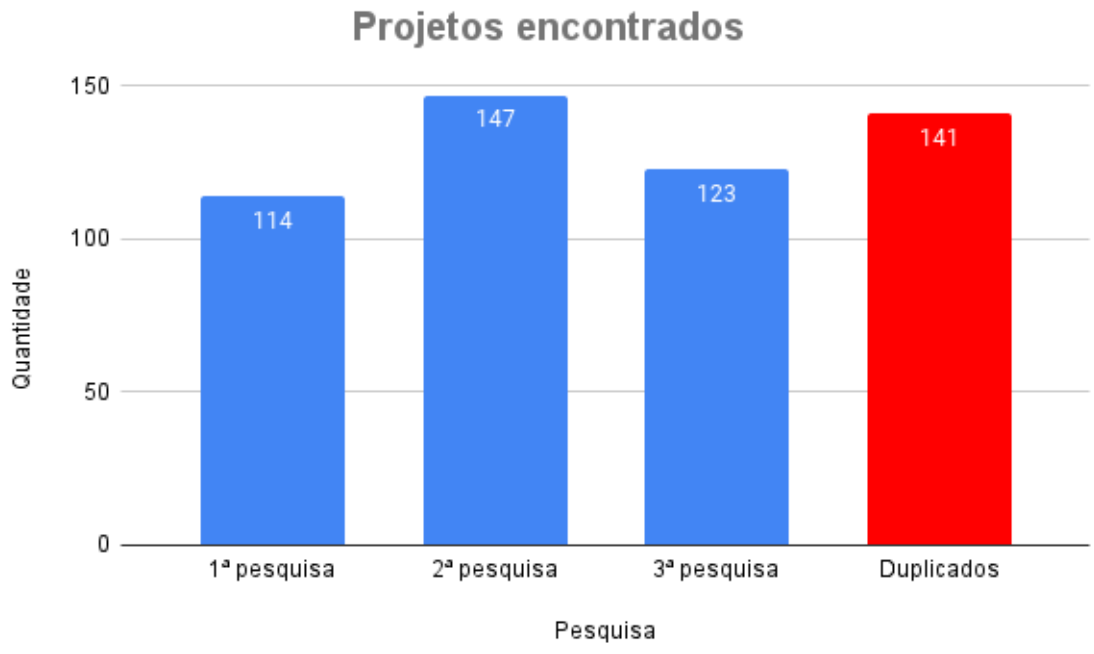

Figura 1. Quantidade de projetos encontrados e removidos

3.1. Linguagens utilizadas nos projetos

Nos projetos aonde foram indicadas as linguagens para desenvolvimento de soluções voltadas à análise de vulnerabilidades de código fonte, as mais utilizadas são, respectivamente, Python (30,7%), Javascript (11,5%) e Java (8,3%), conforme demonstrado na Figura 2:

Figura 2. Linguagens de programação utilizadas nos projetos

3.2. Tipos de licenças dos projetos

Os tipos de licenciamento mais utilizados e representativos nas amostras coletadas foram, respectivamente, MIT License[4] (49,5%), GPL-3.0 License[5] (23,8%) e Apache 2.0 License[6] (17,1%), conforme observado na Figura 3:

[4]MIT License é um tipo de licença permissiva curta e simples com condições que exigem apenas a preservação de direitos autorais e avisos de licença. Obras licenciadas, modificações e obras maiores podem ser distribuídas sob diferentes termos e sem código-fonte.

[5]GNU General Public License v3.0 (GPL-3.0 License) é um tipo de licença de copyleft forte, estando condicionadas à disponibilização do código fonte completo das obras licenciadas e modificações, que incluem obras maiores usando uma obra licenciada, sob a mesma licença. Os avisos de direitos autorais e de licença devem ser preservados. Os colaboradores fornecem uma concessão expressa de direitos de patente.

[6]Apache 2.0 License é um tipo de licença permissiva cujas principais condições exigem a preservação dos direitos autorais e dos avisos de licença. Os contribuidores fornecem uma concessão expressa de direitos de patente. Obras licenciadas, modificações e obras maiores podem ser distribuídas sob diferentes termos e sem código-fonte.

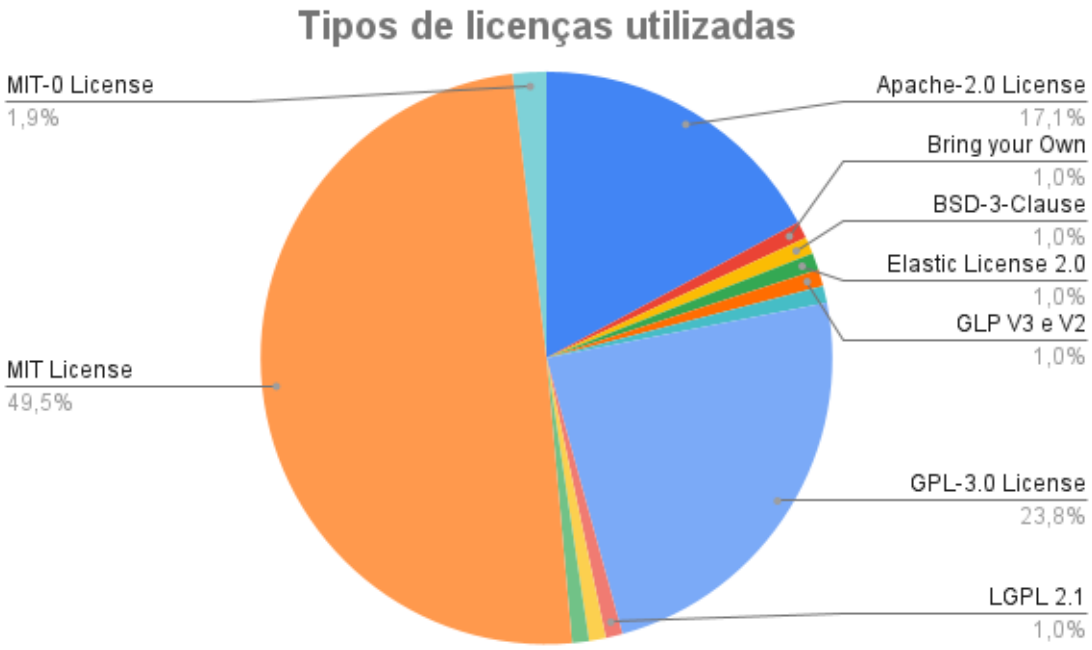

Figura 3. Tipos de licenças utilizadas nos projetos

3.3. Publicações de projetos por ano

Observamos um aumento constante do número de aplicações criadas com o objetivo de melhorar a segurança dos softwares desenvolvidos, representando um incremento médio de 27,6% por ano, a partir de 2016, conforme observado na Figura 4:

Figura 4. Publicações de projetos por ano

3.4. Banco de dados usados nos projetos

Apesar da coleta de dados ter sido realizada, entendemos que os mesmos são irrelevantes, pois não há referência aos bancos de dados utilizados no material analisado em mais de 94% da amostra, conforme a Tabela 3 abaixo.

Tabela 3. Bancos de dados

BANCO DE DADOS	QUANTIDADE
ChatGPT	1
CVE	2
CVE e CWE	1
CVE NVD	1
CVE-2021-45046	1
CWE	1
Metasploit	1
N/A	200
OWASP	2
OWASP e CWE	1
OWASP, CWE, WS-Attacks e CERT	1
Total geral	**213**

3.5. Pontuação dos projetos

Observamos que as pontuações dos projetos foram muito dispersas, variando de 0 a 2096 pontos, tendo o desvio padrão no valor de 26,87, sendo que a grande maioria dos projetos obteve, pelos critérios definidos, entre 0(zero) e 2(dois) pontos, conforme se infere da Figura 5:

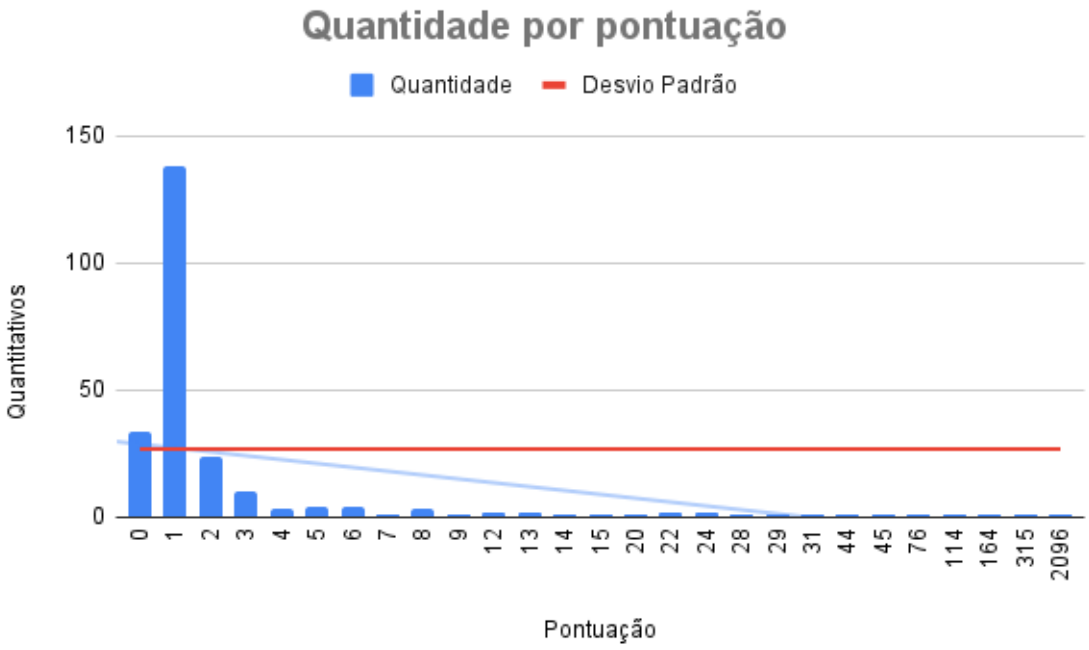

Figura 5. Quantidade de projetos por pontuação

Verificamos também, segmentando-se as 10 (dez) menores pontuações entre os anos 2016 e 2023, que os projetos mais recentes possuem uma pontuação menor, o que faz sentido se analisarmos o tempo de disponibilização dos mesmos ao acesso público, conforme se observa na Figura 6:

Figura 6. Pontuação por ano dos projetos

3.6. Apresentação dos trabalhos selecionados e respostas às questões de pesquisa

Efetivadas todas as análises e aplicados os filtros indicados na Seção 2, Seção 4, os seguintes projetos foram considerados aptos para análise e possível adoção no projeto, conforme disponibilizado nas planilhas do projeto do Github e apresentados resumidamente a seguir:

- **1º) Projeto**: github/codeql[7], ano: 2022, pontuação: 2096
- **2º) Projeto**: Bearer/bearer[8], ano: 2022, pontuação: 315
- **3º) Projeto**: marcinguy/betterscan-ce[9], ano: 2021, pontuação: 76
- **4º) Projeto**: security-code-scan/security-code-scan[10], ano: 2017, pontuação: 45
- **5º) Projeto**: cycodehq-public/cycode-cli[11], ano: 2022, pontuação: 44
- **6º) Projeto**: bridgecrewio/checkov-action[12], ano: 2020, pontuação: 29

Quanto às questões de pesquisa 2.2, obtivemos as seguintes respostas:

- **RQ1**: Não foi possível obter a informação, uma vez prejudicado o avanço na pesquisa dos artigos selecionados, em razão do prazo para elaboração do presente documento;
- **RQ2**: Há uma diversidade de aplicações que fazem pesquisa de vulnerabilidade em código fonte hospedados em repositórios como o Github e Gitlab, bem como nos mais diversos tipos de vulnerabilidades, desde senhas escritas no código, erros de codificação (CWE-Common Weakness Enumeration), exposição e vulnerabilidades comunns (CVE-Common Vulnerabilities and Exposures), bem como outras técnicas SAST-Static Application Security Testing;
- **RQ3** e subconsultas: é respondida pelas informações dispostas nas planilhas do projeto no GitHub, onde destacamos que os quesitos **RQ3.4**, **RQ3.5** e **RQ3.6** tiveram suas respostas prejudicadas pela falta ou alimentação parcial dos dados por parte dos autores dos projetos selecionados.

4. Considerações finais

Os resultados obtidos através da pesquisa e mapeamento de soluções para análise de vulnerabilidade de código demonstram a viabilidade de utilização de uma, ou a combinação de várias soluções, com foco em centralização e automação do monitoramento das etapas do ciclo de vida de desenvolvimento de *software*. O desenvolvimento de novas soluções para pesquisa de vulnerabilidades talvez seja inviável porque, conforme demonstrado no estudo, já há muitas soluções no mercado, sejam elas livres ou proprietárias, não fazendo sentido envidar esforços para reinventar soluções já existentes, salvo se surgir algo muito novo no mercado que ainda não seja atendido pelas soluções existentes.

O estudo demonstrou, ainda, que as soluções não se conversam entre si, havendo uma lacuna entre a aplicação prática dos softwares de análise de vulnerabilidade e os processos relacionados ao ciclo de vida de desenvolvimento de software, em especial às práticas relacionadas ao DevSecOps.

[7]https://github.com/github/codeql
[8]https://github.com/Bearer/bearer
[9]https://github.com/marcinguy/betterscan-ce
[10]https://github.com/security-code-scan/security-code-scan
[11]https://github.com/cycodehq-public/cycode-cli
[12]https://github.com/bridgecrewio/checkov-action

4.1. Principais contribuições

O trabalho contribui como fonte de pesquisa para aqueles que buscam desenvolver soluções de software voltadas à segurança da informação, ou simplesmente desenvolver softwares mais seguros, além de poder ser utilizado como base para outras pesquisas na mesma linha.

4.2. Limitações

Considerando que parte da pesquisa foi prejudicada em razão da exiguidade do tempo disponível para realização dos trabalhos, há uma chance considerável de os resultados da pesquisa terem sido afetados pela falta de direcionamento trazido pela análise do que é atualmente recomendado sobre análise de vulnerabilidades, conforme a questão de pesquisa **RQ1**, que poderia modificar completamente os argumentos utilizados nas buscas pelas aplicações. Foram identificados vários projetos fazendo referências a soluções com base em softwares proprietários, que possuem certas limitações que talvez pudessem ser exploradas como viáveis, mas não foram prospectadas.

4.3. Trabalhos futuros

Conforme demonstrado anteriormente, talvez seja interessante complementar o presente trabalho com o término da análise e resposta à questão de pesquisa **RQ1**, bem como proceder um maior aprofundamento no uso de soluções proprietárias e o impacto das limitações do uso sem aquisição de licenças na efetividade do seu funcionamento.

5. Referência bibliográfica

Referências

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.

Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., and Whangbo, T. K. (2022). Critical challenges to adopt devops culture in software organizations: A systematic review. *IEEE Access*, 10:14339–14349.

Luz, W. P., Pinto, G., and Bonifácio, R. (2019). Adopting devops in the real world: A theory, a model, and a case study. *Journal of Systems and Software*, 157:110384.