Análise de Código para Detecção de Vulnerabilidades: Comparação entre Abordagens Baseadas em Aprendizado de Máquina e em Algoritmos Determinísticos

Ana Carolina | Gustavo Barbosa | João Pedro | Maria Eduarda | Pedro Camara | Wanessa Dias

Contexto

R. Shirey (2007, pag. 333, tradução nossa) define uma vulnerabilidade de software como:

> **"Uma falha ou fragilidade no projeto, implementação ou na operação e gerenciamento do sistema que pode ser explorada para violar a política de segurança do mesmo".**

Essas falhas, mesmo pequenas, podem ter graves consequências, como Mars Polar Lander (MPL) e Deep Space 2 (DS2).

Métodos tradicionais de teste podem consumir tempo e recursos que a organização não possui (ABDULLAH1 *et al.*, 2020) (GAROUS1 *et al.*, 2020).

Sendo assim, surgiram ferramentas de análise de software que identificam falhas automáticamente, gerando um relatório de segurança detalhado.

Problema

Embora existam diversos estudos sobre ferramentas de análise de código, existe uma lacuna na comparação entre ferramentas baseadas em *machine learning* e em algoritmos tradicionais determinísticos, assim, dificultando uma escolha justa, baseada em dados, entre as duas abordagens

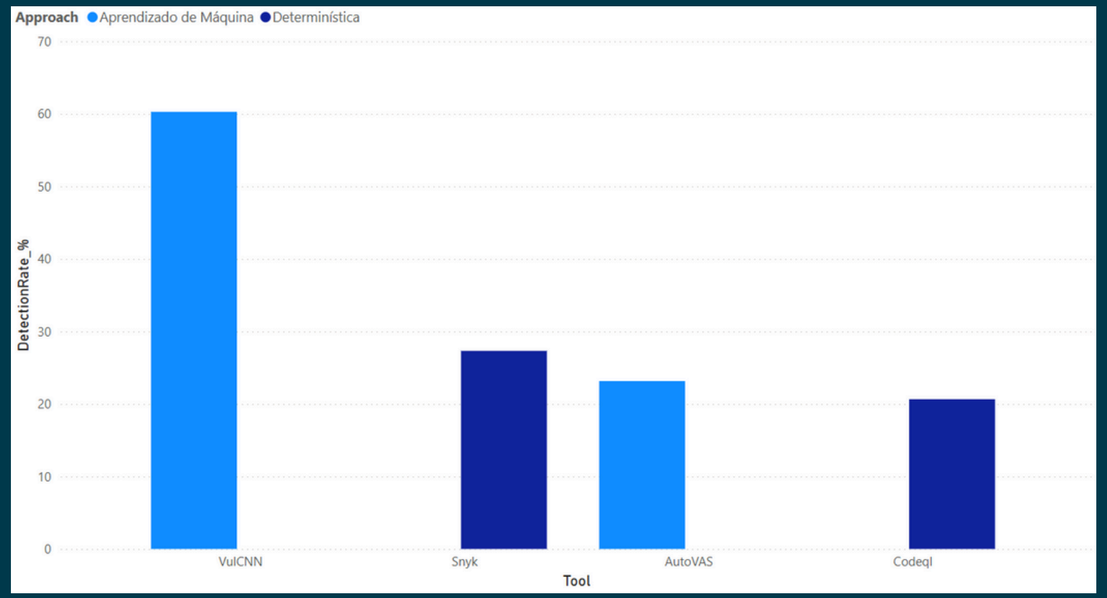
Objetivo

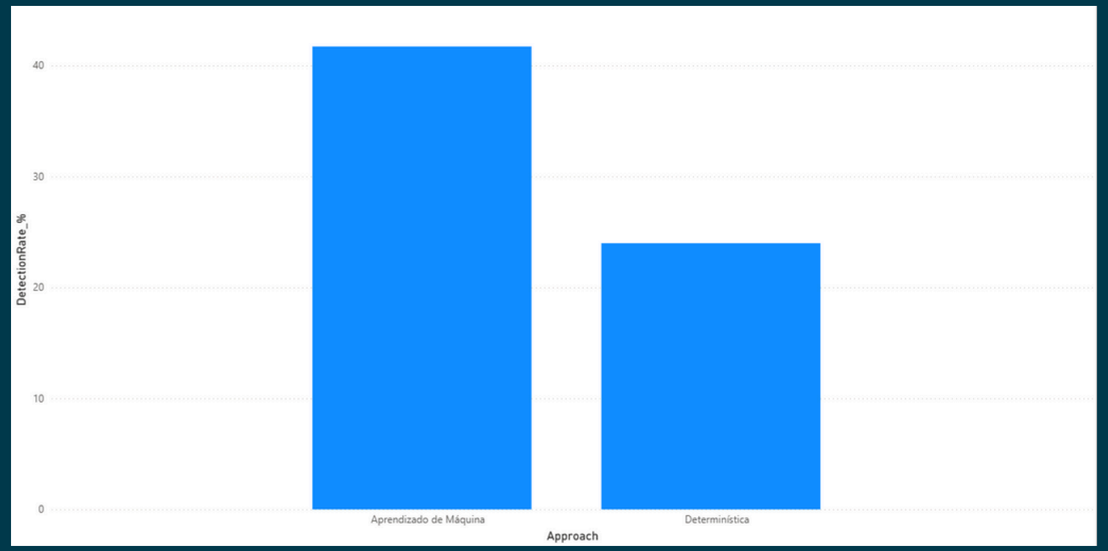
Investigar a eficácia de diferentes abordagens de análise de código, sendo elas as que são baseadas em aprendizado de máquina e em algoritmos tradicionais determinísticos, na detecção de vulnerabilidades

Qual a precisão de ferramentas de análise de código que utilizam aprendizado de máquina em comparação com algoritmos tradicionais determinísticos?

1.1	Taxa de vulnerabilidades encontradas (JEON; KIM, 2021) (ARUSOAIE et al., 2017) (MASKUR; ASNAR, 2019)
1.2	Taxa de falsos positivos e negativos (JEON; KIM, 2021) (ARUSOAIE et al., 2017) (Emanuelsson & Nilsson, 2008) (Russell et al., 2018)
1.3	Detecções únicas de cada ferramenta (ARUSOAIE et al., 2017)

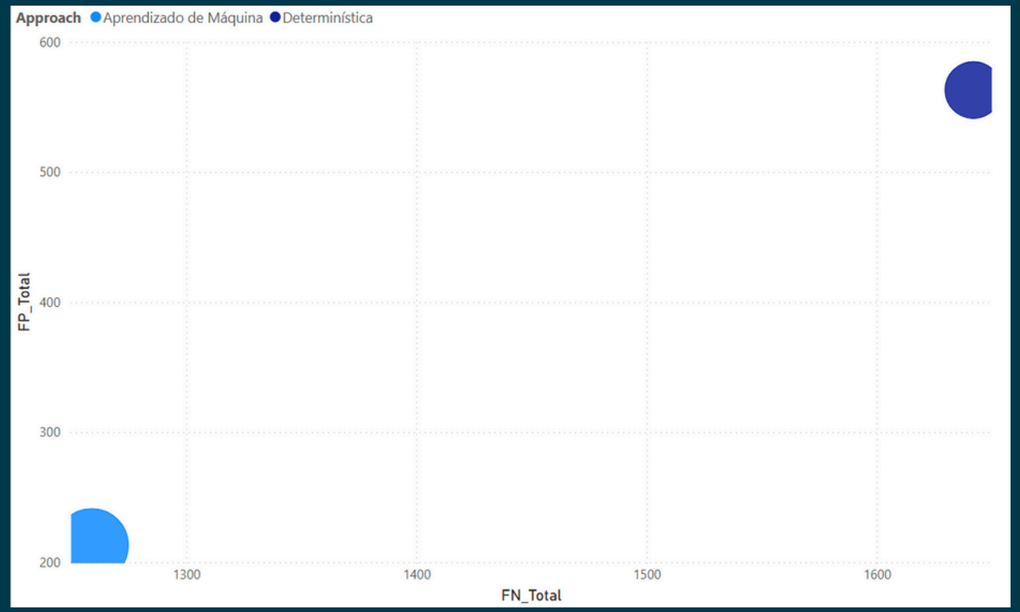
RQ1 - M1

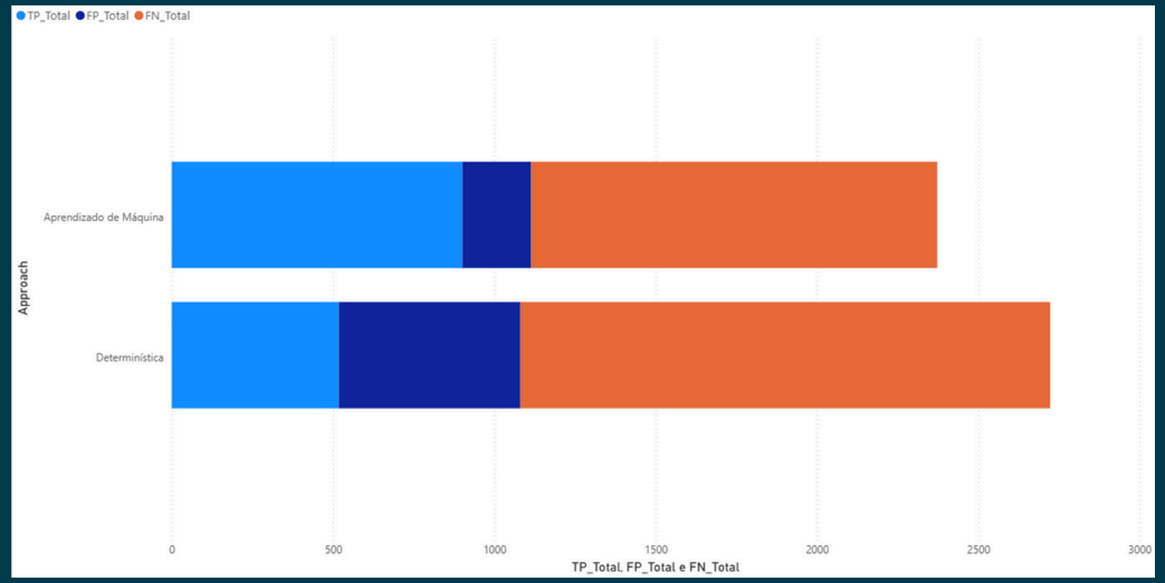




Taxa de vulnerabilidades encontradas por ferramenta/abordagem

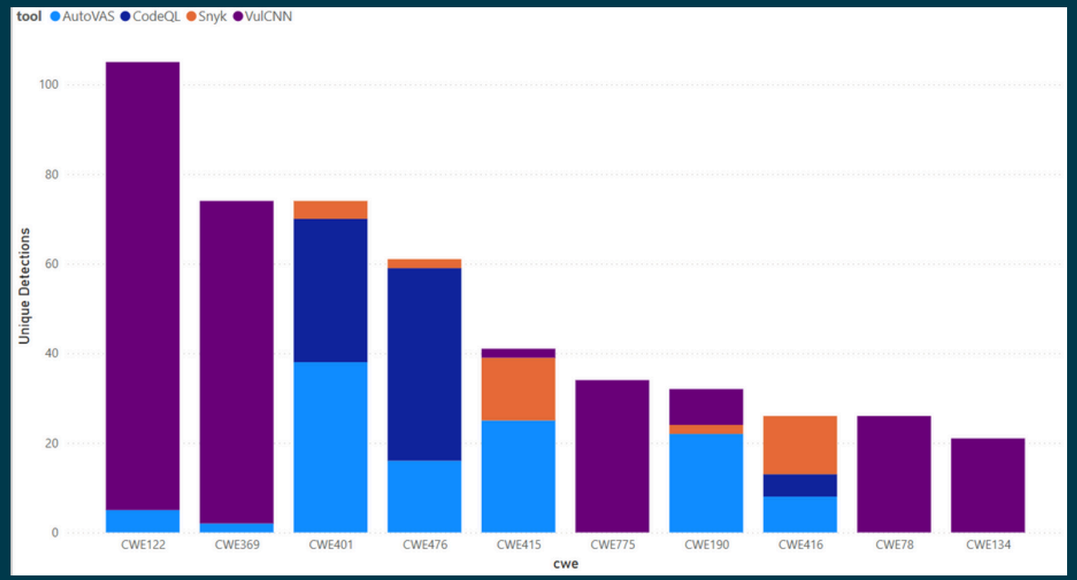
RQ1 - M2

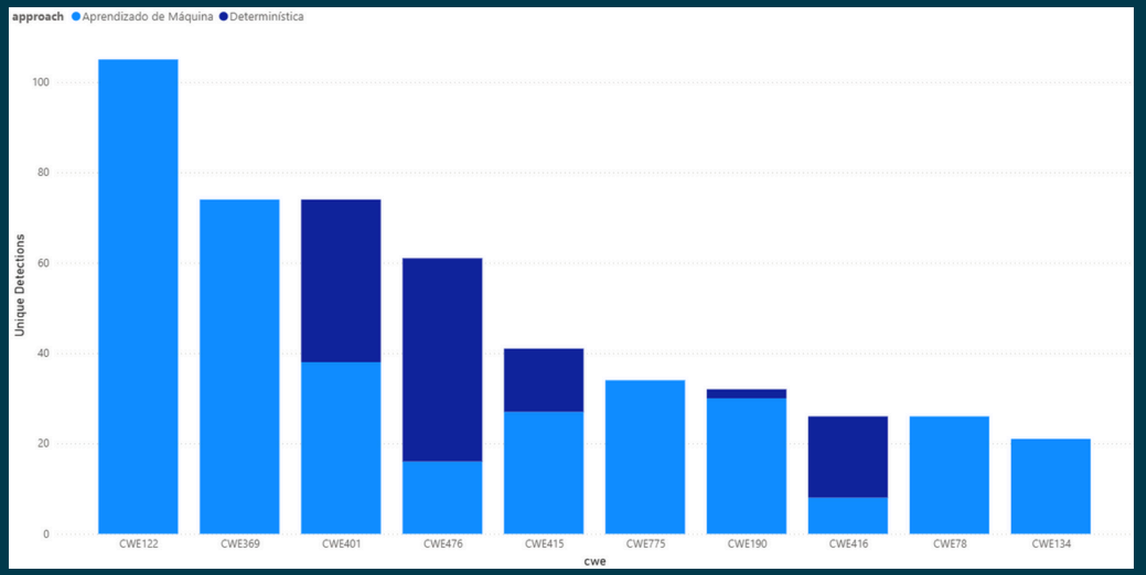




Taxa de falsos positivos/negativos por ferramenta

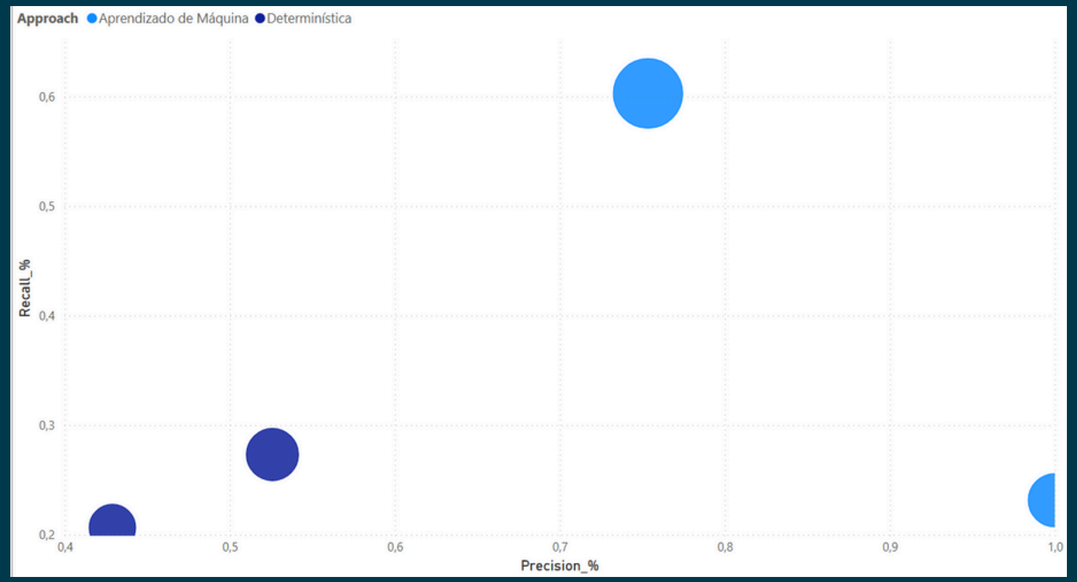
RQ1 - M3

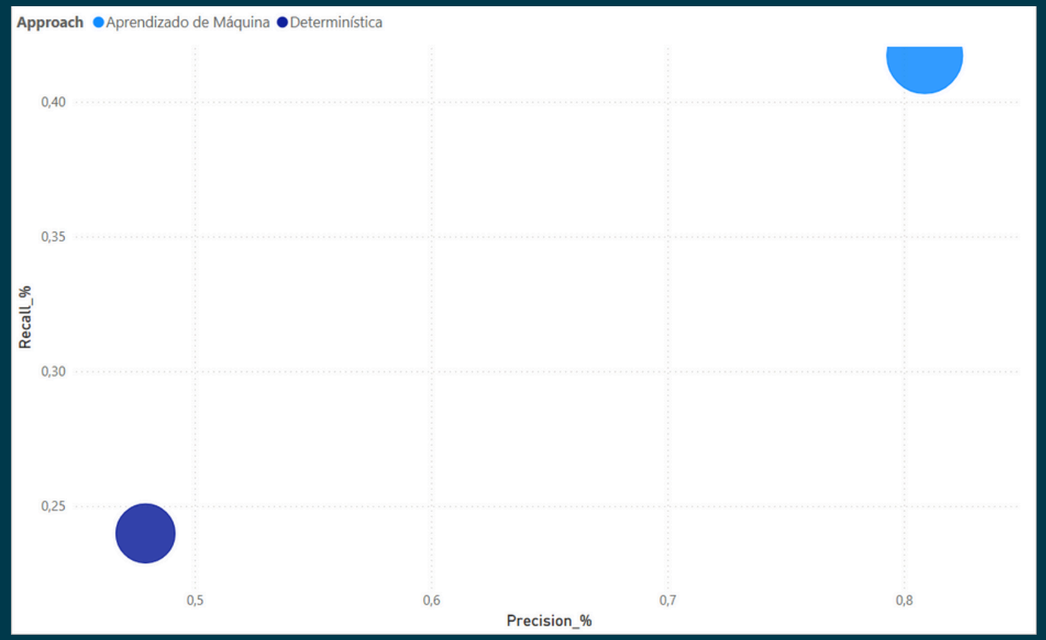




Detecções únicas de cada ferramenta/abordagem por categoria CWE

RQ1



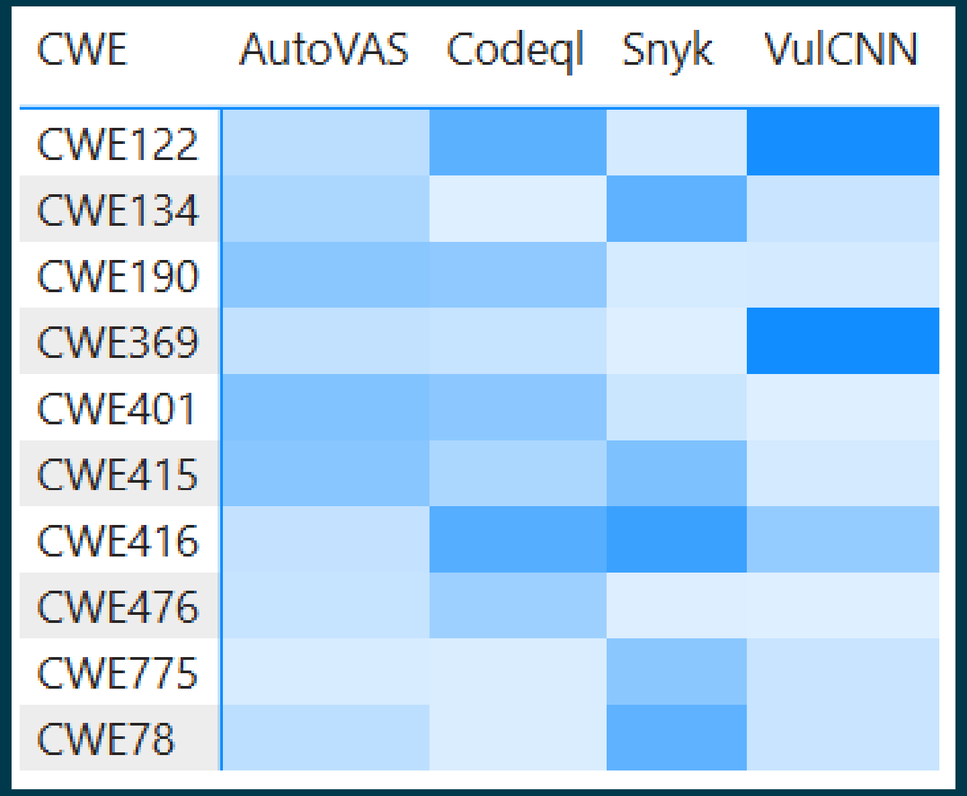


Cobertura comparada com a confiabilidade

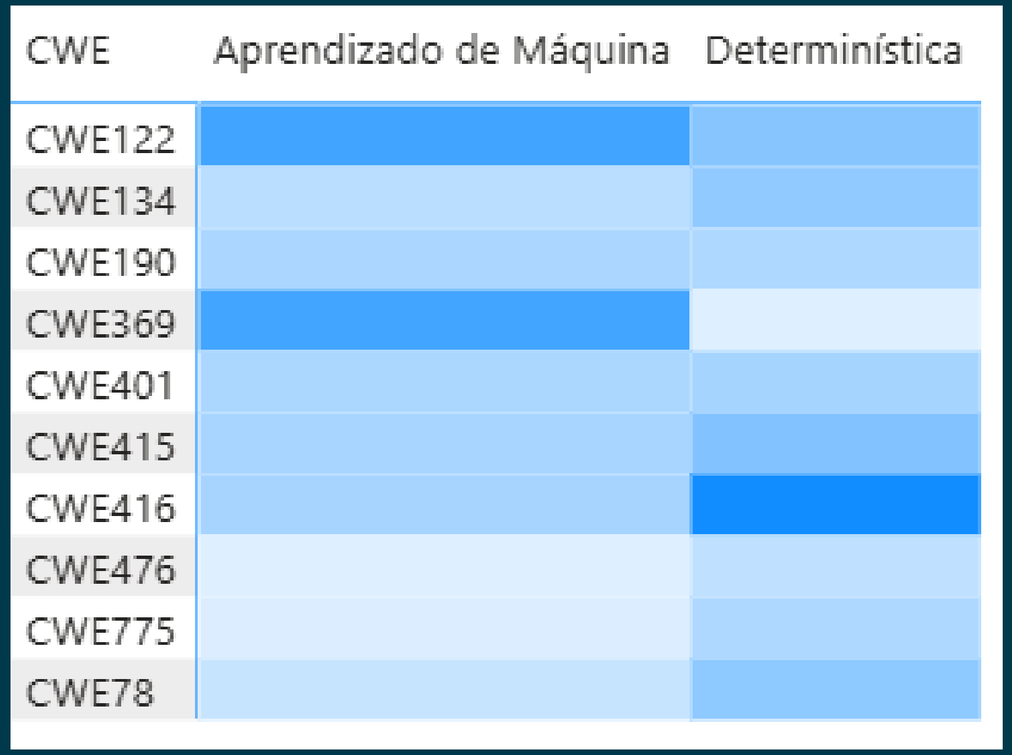
Quais tipos de vulnerabilidades mais detectadas e negligenciadas por cada tipo de abordagem?

2.1	Cobertura por categoria CWE (buffer overflow, uso de variáveis não inicializadas, injeções, entre outros) (MITRE, 2024) (ARUSOAIE et al., 2017) (RUSSELL et al., 2018) (STEENHOEK et al., 2023)
2.2	Vulnerabilidades não detectadas por cada abordagem (pontos cegos) (RUSSELL et al., 2018)
2.3	Vulnerabilidades mais encontradas por cada ferramenta (RUSSELL et al., 2018)

RQ2 - M1



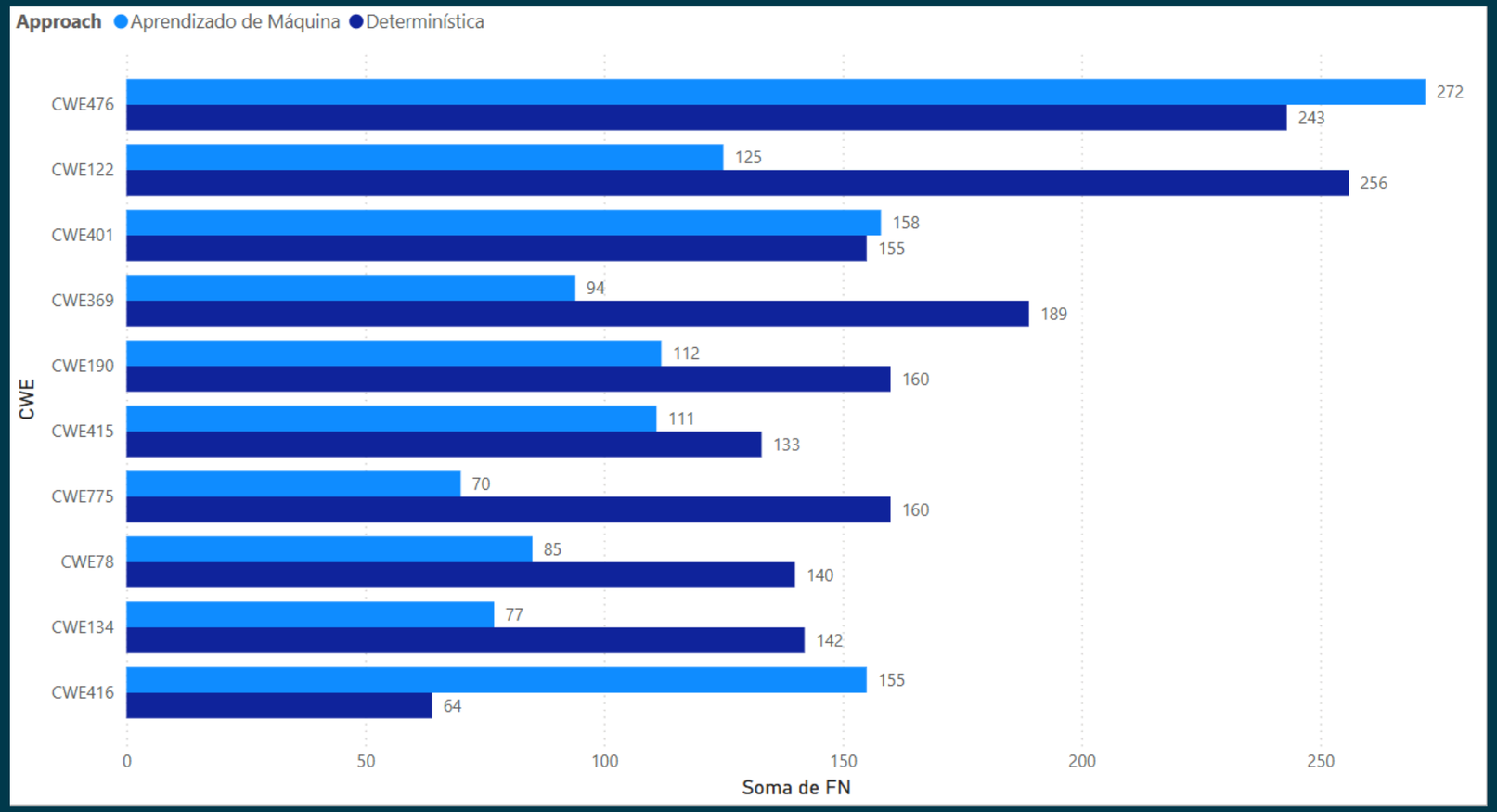
CWE	AutoVAS	Codeql	Snyk	VulCNN
CWE122				
CWE134				
CWE190				
CWE369				
CWE401				
CWE415				
CWE416				
CWE476				
CWE775				
CWE78				



CWE	Aprendizado de Máquina	Determinística
CWE122		
CWE134		
CWE190		
CWE369		
CWE401		
CWE415		
CWE416		
CWE476		
CWE775		
CWE78		

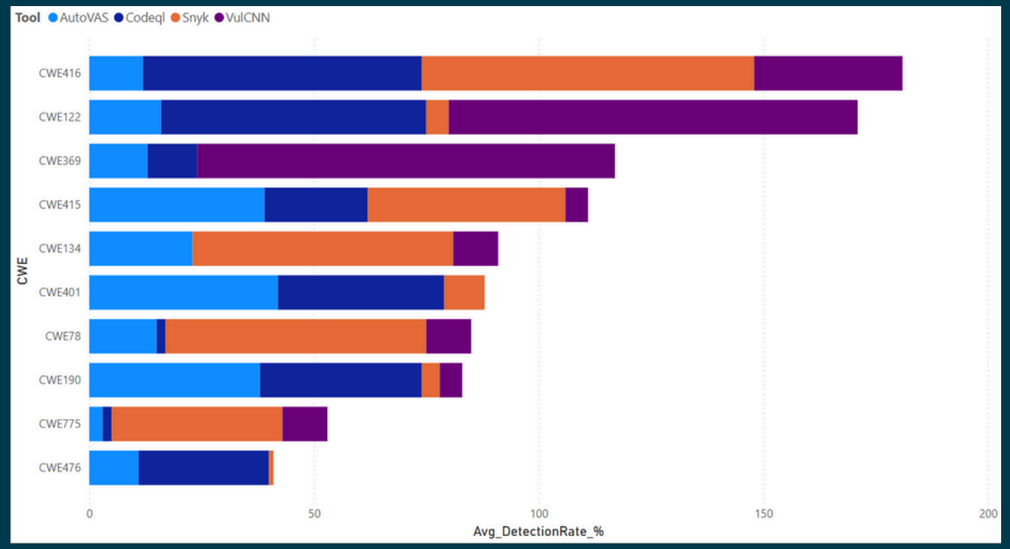
Heatmap de Categoria CWE por ferramenta e abordagem

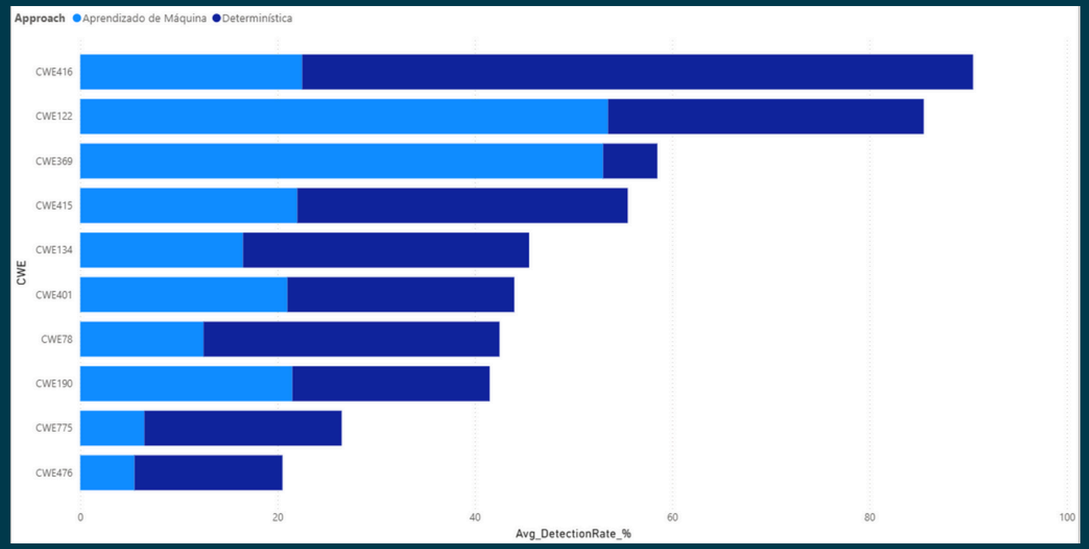
RQ2 - M2



Vulnerabilidades não encontradas por cada ferramenta

RQ2 - M3

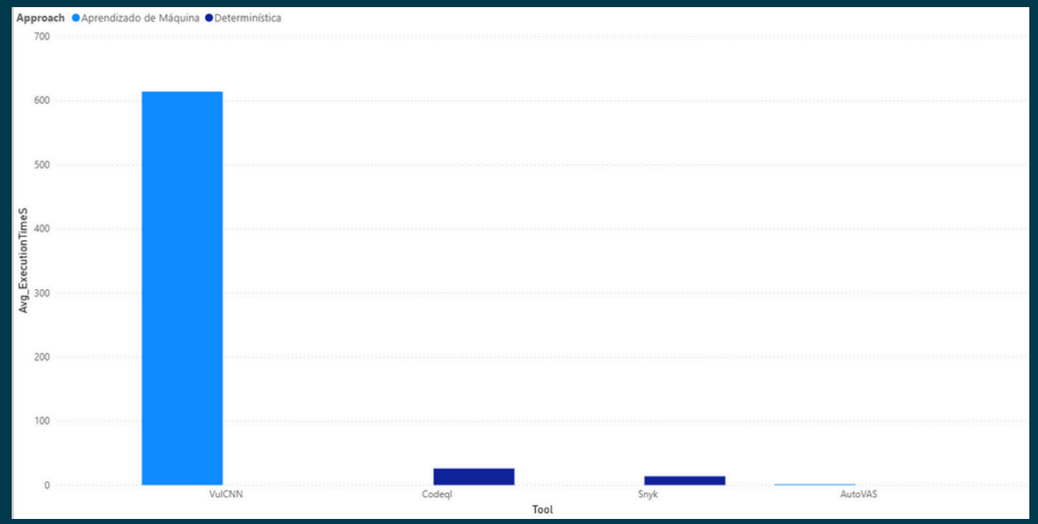


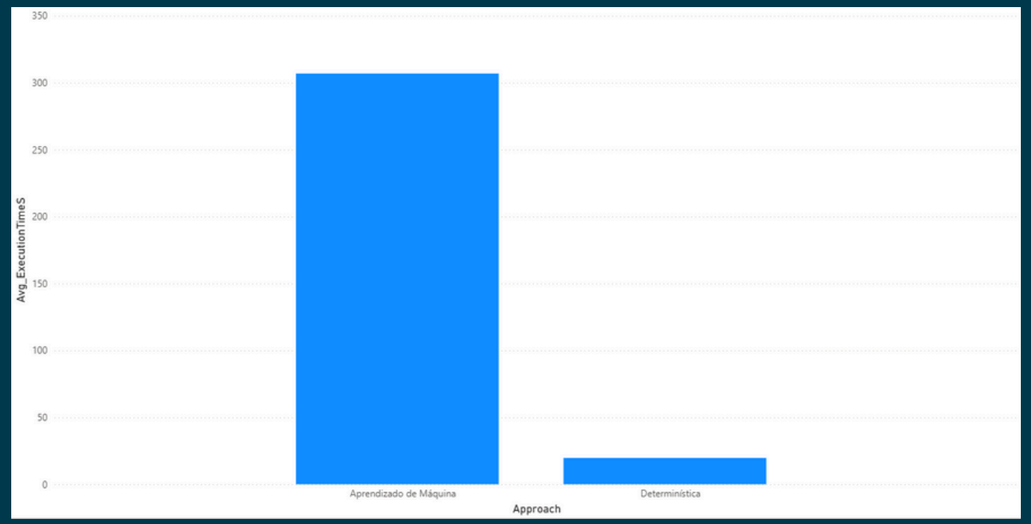


Vulnerabilidades mais encontradas por cada ferramenta/abordagem

Desepenho - SARD

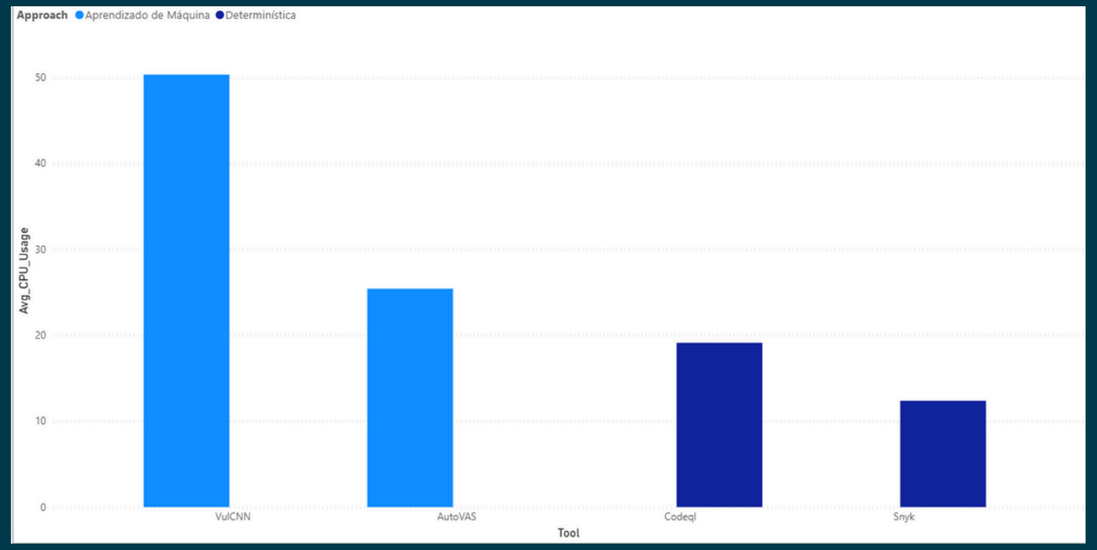
Tempo de Execução

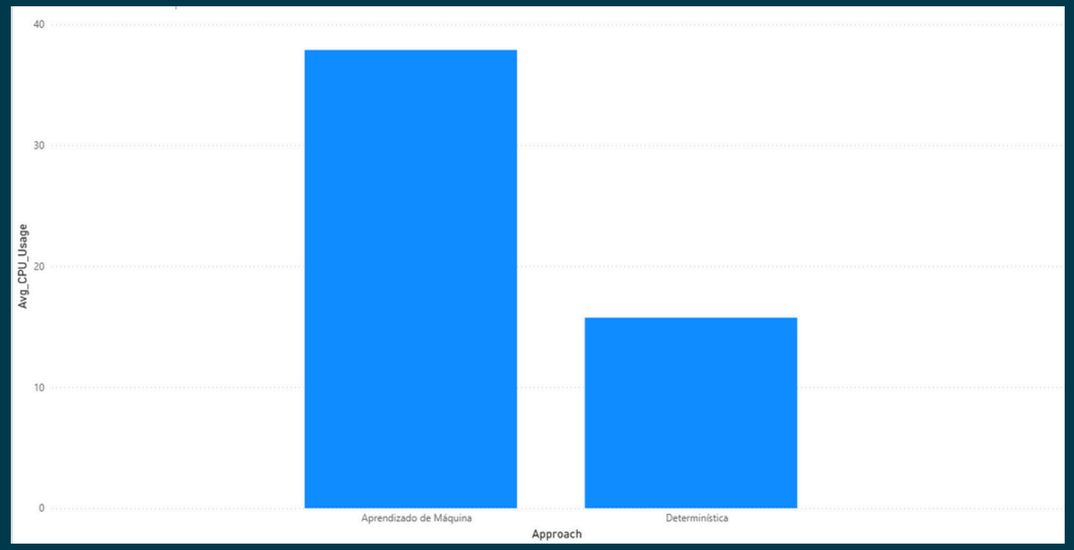




Desepenho - SARD

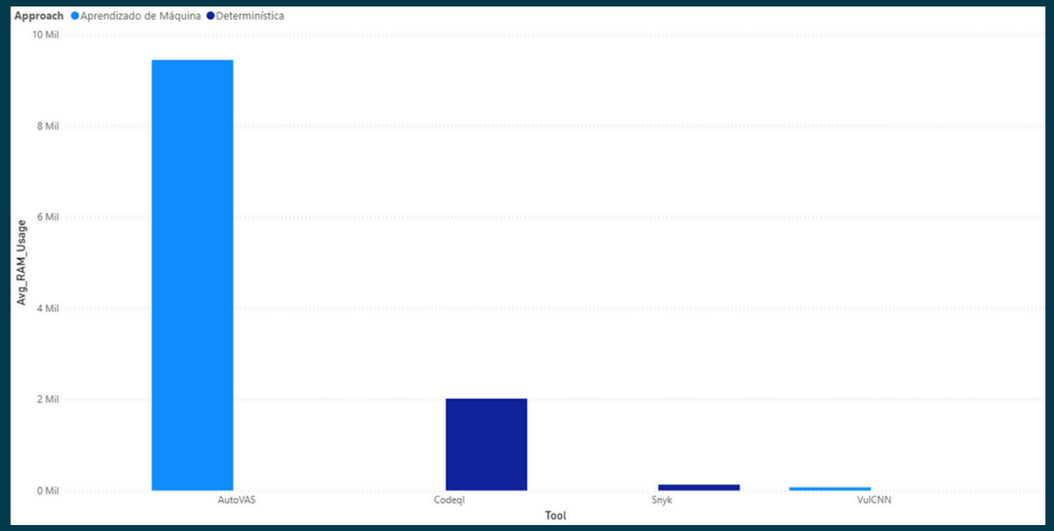
Uso médio de CPU

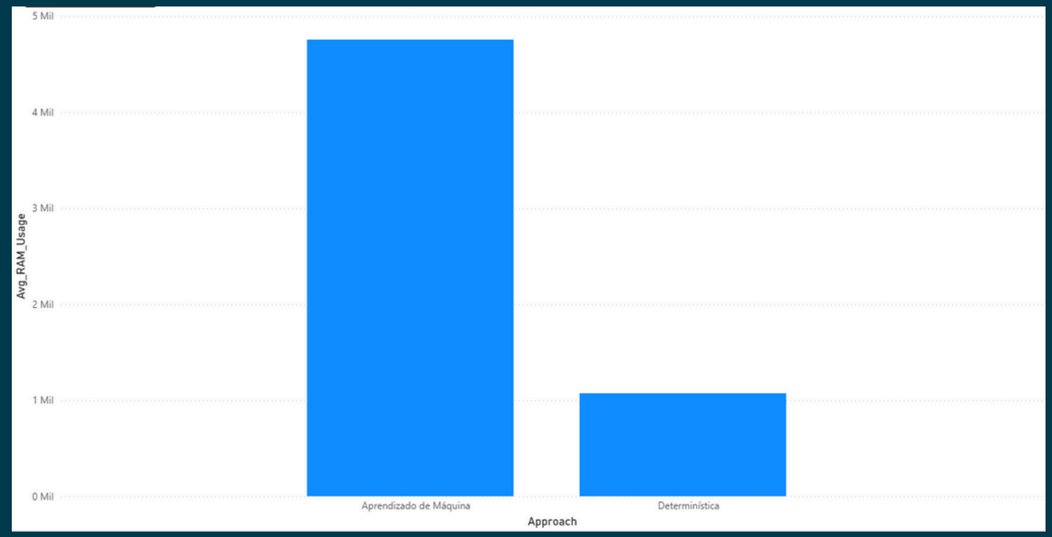




Desepenho - SARD

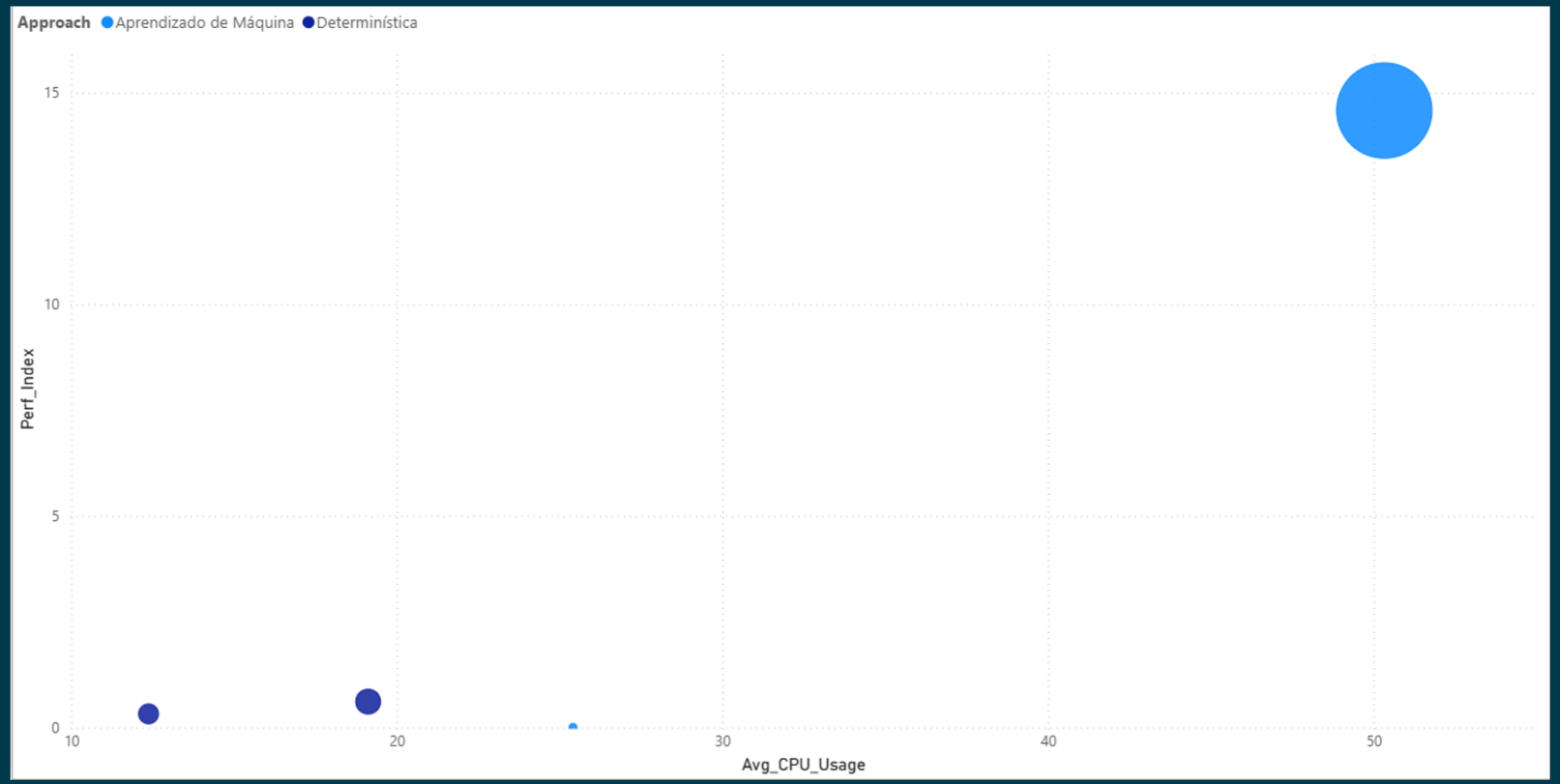
Uso médio de RAM





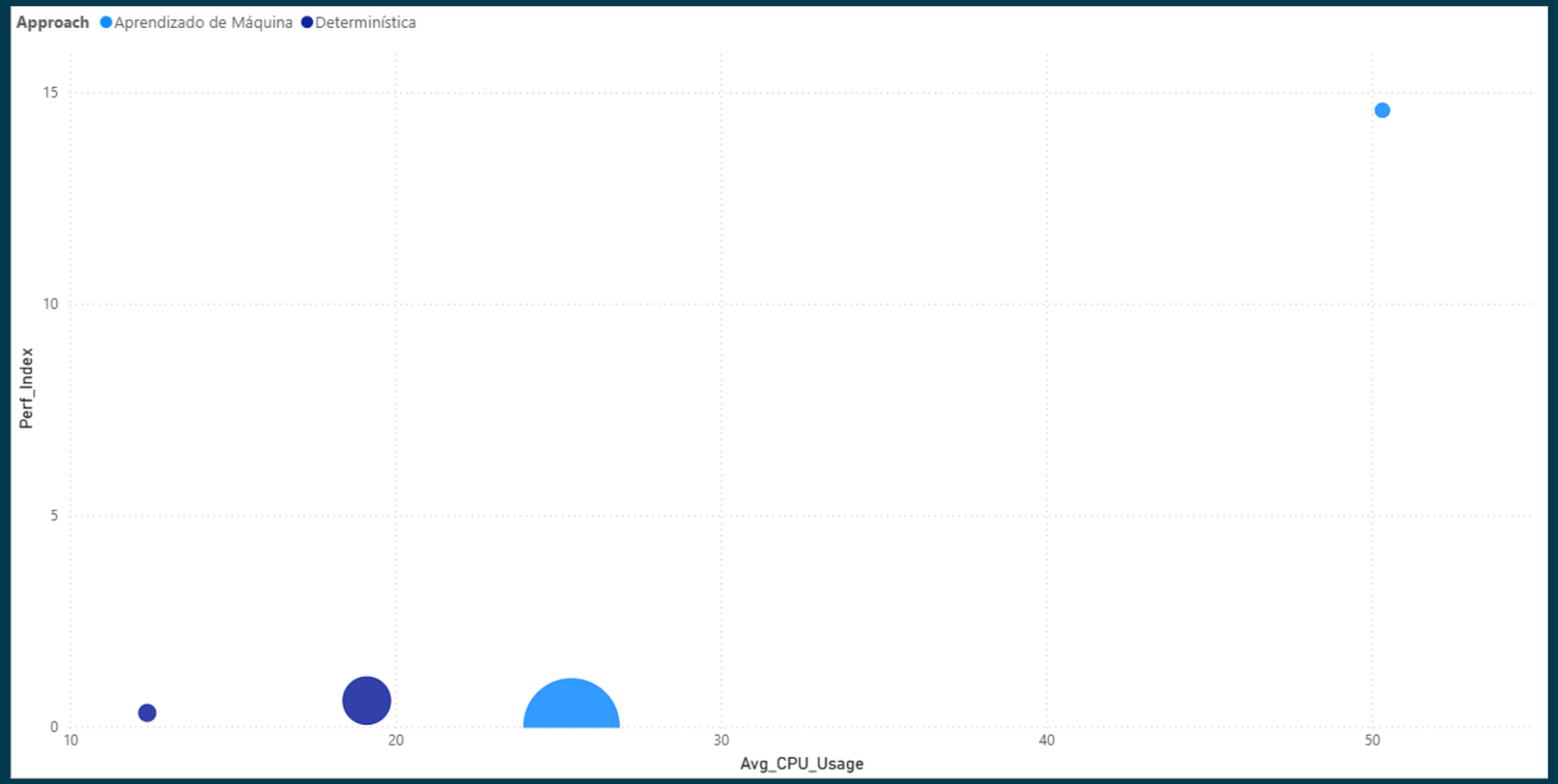
Desepenho - SARD

Precisão X Uso de CPU X Tempo de Execução



Desepenho - SARD

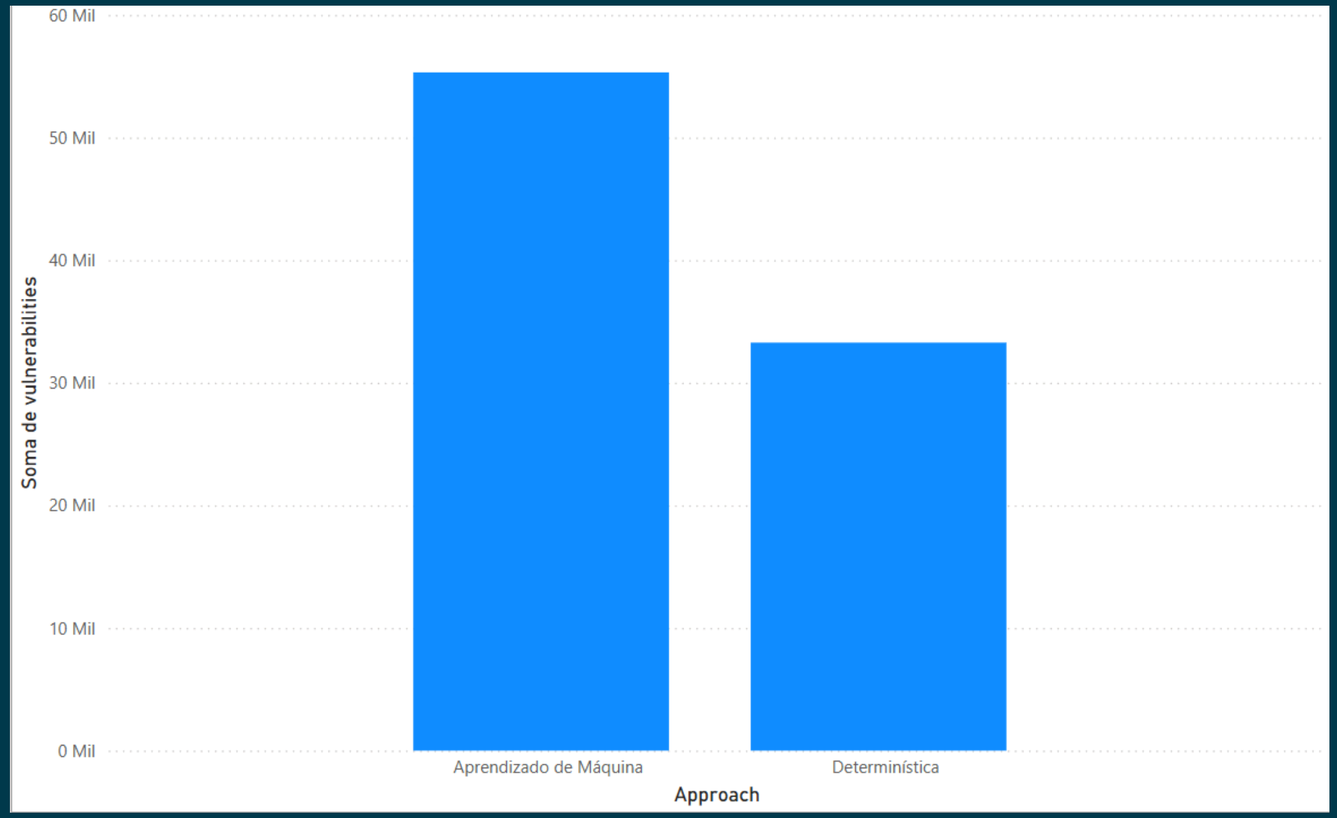
Precisão X Uso de CPU X Uso de RAM



Como as ferramentas de análise se comportam em ambientes reais?

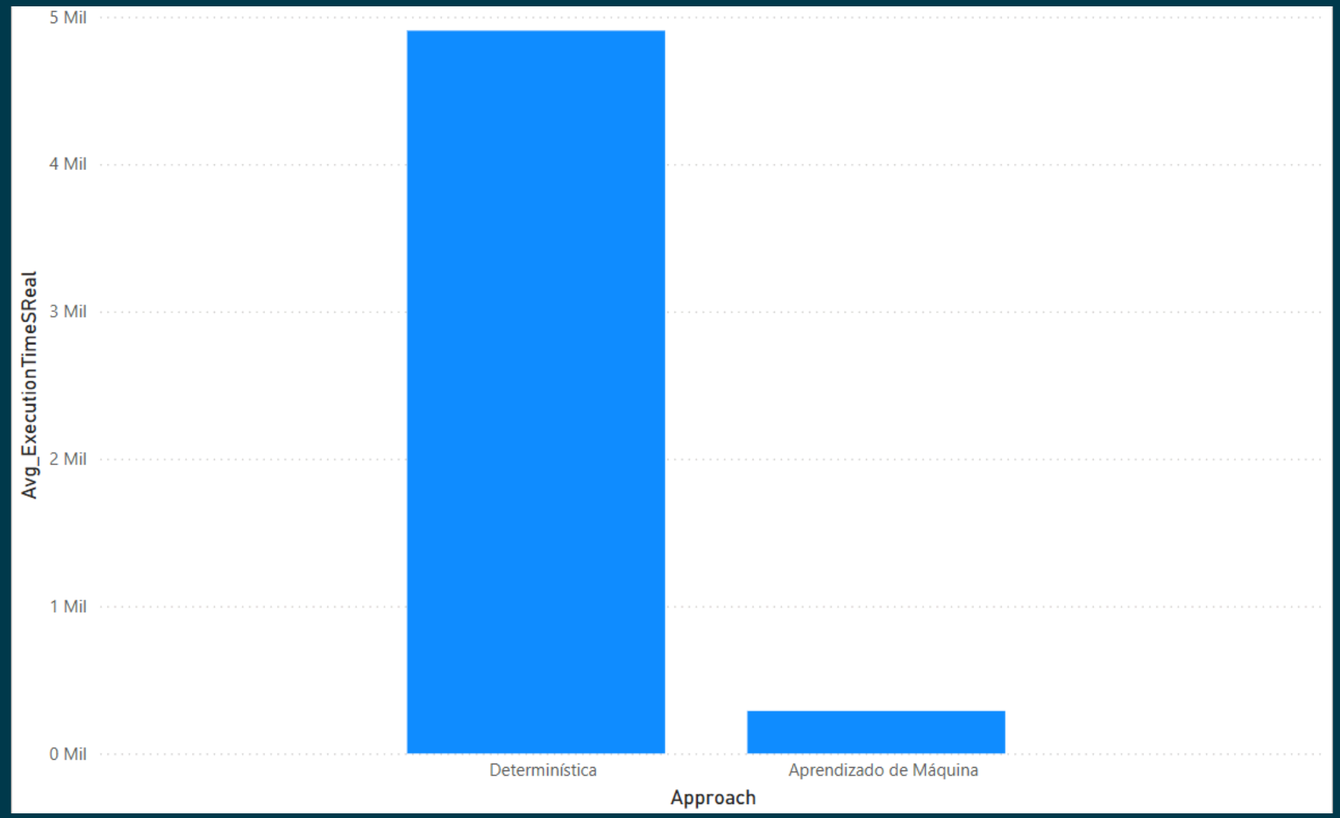
3.1	Quantidade de vulnerabilidades encontradas
3.2	Tempo gasto de análise
3.3	Utilização média dos recursos computacionais (CPU e RAM)

RQ3 - M1



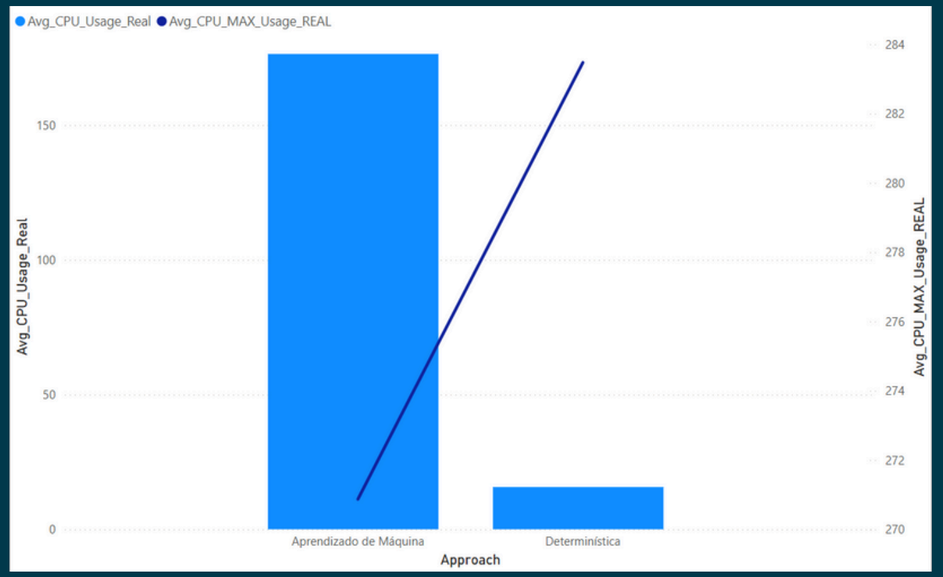
Quantidade de vulnerabilidades encontradas por abordagem

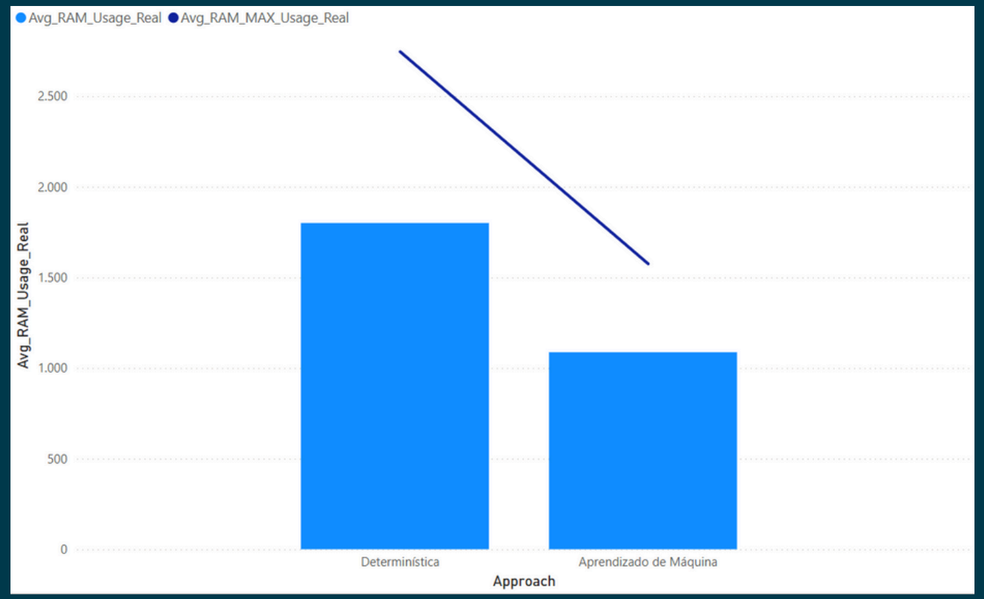
RQ3- M2



Tempo de execução de cada análise

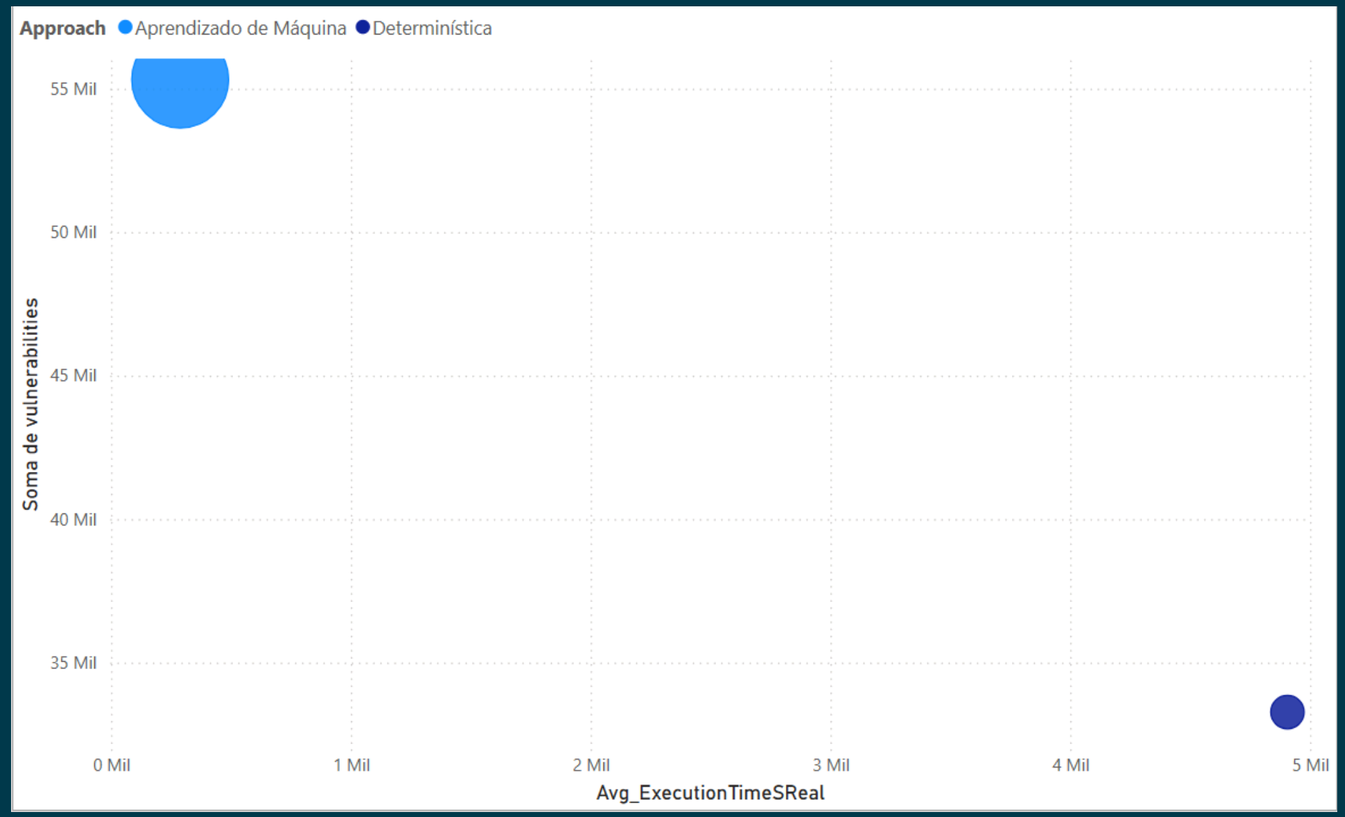
RQ3 - M3





Uso médio dos recursos computacionais de cada abordagem

RQ3



Vulnerabilidades encontradas x tempo de execução x uso médio de cpu

Metodologia

- As ferramentas serão utilizadas nos casos de teste disponibilizados pelo National Institute of Standard and Technology (NIST) no Software Assurance Reference Dataset (SARD) (NIST, [20__] e em alguns repositórios do GitHub (GitHub, [20__])

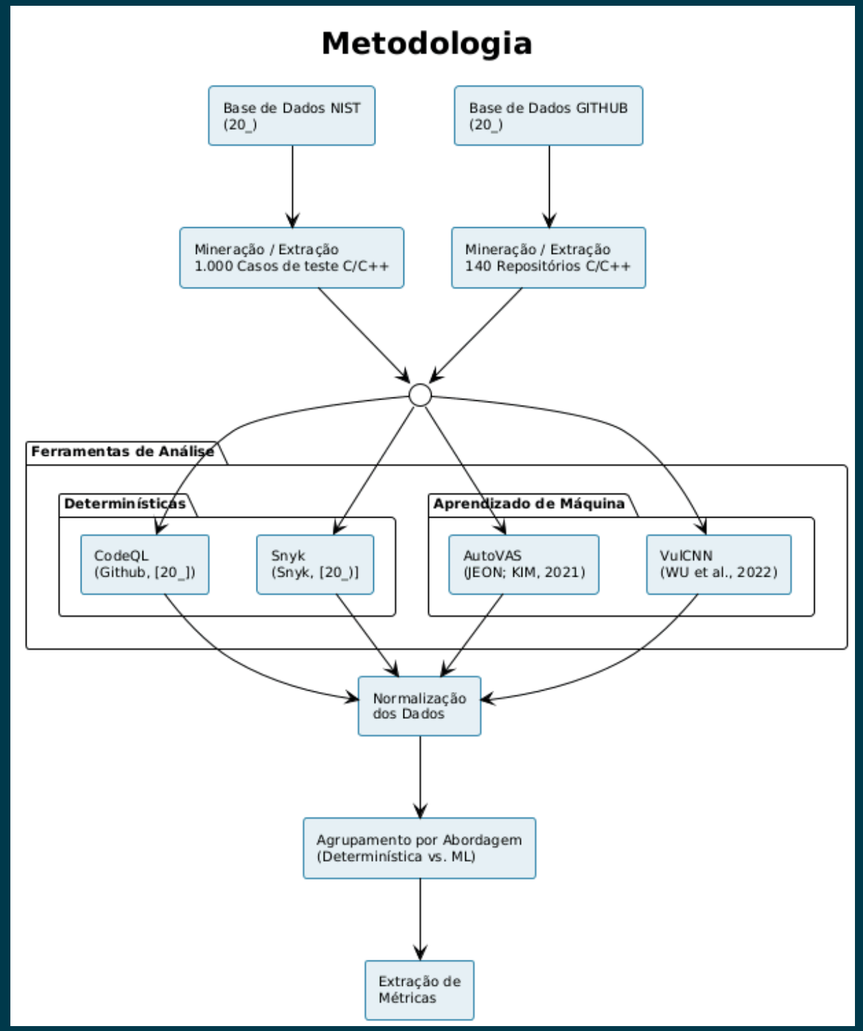
- O NIST SARD é um banco de dados que contém 450.000 casos de teste, todos com suas vunerabilidades identificadas de acordo com o padrão Common Weakness Enumeration

- A linguagem que será analisada é C/C++, visto que é coberta por todas ferramentas.

Metodologia

- Para responder as perguntas aqui levantadas, a pesquisa utiliza as seguintes ferramentas determinísticas:
    - **Codeql** (GITHUB, [20__])
    - **Snyk** (SNYK, [20__])
- E as seguintes ferramentas baseadas em aprendizado de máquina:
    - **AutoVAS** (JEON; KIM, 2021)
    - **VulCNN** (WU et al., 2022)
- A saída das ferramentas é comparada com os dados do banco de dados apresentado para medir as ferramentas.

Metodologia



Referências

ABDULLAHI, Shamsu; ZAKARI, Abubakar; ABDU, Haruna; NURA, Amina; ZAYYAD, Musa Ahmed; SULEIMAN, Salisu; ADAMU, Alhassan; MASHASHA, Abdulfatahu Samaila. Software testing: review on tools, techniques and challenges. International Journal of Advanced Research in Technology and Innovation, v. 2, n. 2, p. 11-18, 2020.

ALBEE, Arden; BATTEL, Steven; BRACE, Richard; BURDICK, Garry; CASANI, John; LAVELL, Jeffrey; LEISING, Charles; MACPHERSON, Duncan; BURR, Peter; DIPPREY, Duane. Report on the loss of the Mars Polar Lander and Deep Space 2 missions. 2000.

ARUSOAIE, A.; CIOBÂCA, S.; CRACIUN, V.; GAVRILUT, D.; LUCANU, D. A comparison of open-source static analysis tools for vulnerability detection in C/C++ code. In: INTERNATIONAL SYMPOSIUM ON SYMBOLIC AND NUMERIC ALGORITHMS FOR SCIENTIFIC COMPUTING (SYNASC), 19., 2017, Timisoara. Proceedings.... Timisoara: IEEE, 2017. p. 161-168. DOI: https://doi.org/10.1109/SYNASC.2017.00035.

EMANUELSSON, Pär; NILSSON, Ulf. A comparative study of industrial static analysis tools. Electronic Notes in Theoretical Computer Science, v. 217, p. 5-21, 2008. DOI: https://doi.org/10.1016/j.entcs.2008.06.039.

GAROUSI, Vahid; FELDERER, Michael; KUHRMANN, Marco; HERKILOĞLU, Kadir; ELDH, Sigrid. Exploring the industry's challenges in software testing: an empirical study. *Journal of Software: Evolution and Process*, v. 32, n. 8, e2251, 2020. DOI: https://doi.org/10.1002/smr.2251.

JEON, Sanghoon; KIM, Huy Kang. AutoVAS: An automated vulnerability analysis system with a deep learning approach. Computers & Security, v. 106, 2021. DOI: https://doi.org/10.1016/j.cose.2021.102308.

LEVENSON, Nancy G.; TURNER, Clark S. An investigation of the Therac-25 accidents. Computer, v. 26, n. 7, p. 18-41, July 1993. DOI: https://doi.org/10.1109/MC.1993.274940.

WU, Yueming; ZOU, Deqing; DOU, Shihan; YANG, Wei; XU, Duo; JIN, Hai. VulCNN: an image-inspired scalable vulnerability detection system. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 44., 2022, Pittsburgh, PA. Proceedings... New York: Association for Computing Machinery, 2022. p. 2365–2376. DOI: 10.1145/3510003.3510229.

MASKUR, A. F.; ASNAR, Y. D. W. Static code analysis tools with the taint analysis method for detecting web application vulnerability. In: INTERNATIONAL CONFERENCE ON DATA AND SOFTWARE ENGINEERING (ICoDSE), 2019, Pontianak. Proceedings.... Pontianak: IEEE, 2019. p. 1-6. DOI: https://doi.org/10.1109/ICoDSE48700.2019.9092614.

MITRE. CWE—2024 CWE Top 25 Most Dangerous Software Weaknesses. Disponível em: https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST). Software Assurance Reference Dataset (SARD). [20__]. Disponível em: https://samate.nist.gov/SARD/.

Qadir S, Waheed E, Khanum A, Jehan S. 2025. Comparative evaluation of approaches & tools for effective security testing of Web applications. PeerJ Computer Science 11:e2821 https://doi.org/10.7717/peerj-cs.2821.

RUSSELL, R. et al. Automated vulnerability detection in source code using deep representation learning. In: IEEE INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS (ICMLA), 17., 2018, Orlando. Proceedings.... Orlando: IEEE, 2018. p. 757-762. DOI: https://doi.org/10.1109/ICMLA.2018.00120.

SÁNCHEZ, Mario Calín; GEa, Juan Manuel Carrillo de; FERNÁNDEZ-ALEMÁN, José Luis; GARCERÁN, Jesús; TOVAL, Ambrosio. Software vulnerabilities overview: A descriptive study. *Tsinghua Science and Technology*, v. 25, n. 2, p. 270-280, abr. 2020. DOI: https://doi.org/10.26599/TST.2019.9010003.

SHIREY, R. RFC 4949: Internet Security Glossary, Version 2. Fremont: IETF, 2007. 372 p. Disponível em: https://www.rfc-editor.org/rfc/rfc4949. Acesso em: 24 ago. 2025.

SNYK. Snyk — Segurança de código, dependências, contêineres e infraestrutura. [20__]. Disponível em: https://snyk.io/pt-BR/.

STEENHOEK, B.; RAHMAN, M. M.; JILES, R.; LE, W. An empirical study of deep learning models for vulnerability detection. In: IEEE/ACM INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING – ICSE, 45., 2023, Melbourne. Proceedings... Melbourne: IEEE, 2023. p. 2237-2248. DOI: https://doi.org/10.1109/ICSE48619.2023.00188.

GITHUB. CodeQL — Descubra vulnerabilidades em toda a base de código. 2021. Disponível em: https://codeql.github.com/. Acesso em: 22 out. 2025.

WU, F.; WANG, J.; LIU, J.; WANG, W. Vulnerability detection with deep learning. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER AND COMMUNICATIONS – ICCC, 3., 2017, Chengdu. Proceedings... Chengdu: IEEE, 2017. p. 1298-1302. DOI: https://doi.org/10.1109/CompComm.2017.8322752.

GITHUB, Inc. GitHub: where the world builds software. Disponível em: https://github.com/. Acesso em: 11 nov. 2025.