OTH REGENSBURG

Automated Vulnerability Scanner for the Cyber Resilience Act

Sandro Falter, Gerald Brukh, Max Wess and
Sebastian Fischer (OTH Regensburg)

CLOUD COMPUTING 2024 - April 14, 2024 to April 18, 2024 - Venice, Italy

OTH REGENSBURG

Prof. Dr. Sebastian Fischer

since 2023	Professor for "Computer Science and System Security" at OTH Regensburg, Germany
2021 - 2023	Lecturer at OTH Regensburg, Germany
2022	Dr. rer. nat. at Freie Universität Berlin, Germany
2018 - 2021	Research Associate at Fraunhofer AISEC, Germany
2015 - 2018	Research Associate at OTH Regensburg, Germany
2015	M. Sc. Applied Research in Computer Science, OTH Regensburg
2013	B. Sc. Computer Science, OTH Regensburg

sebastian.fischer@oth-regensburg.de

Contents

OTH REGENSBURG

Cyber Resilience Act (CRA)

- Introduces new mandatory cybersecurity requirements for hardware and software products throughout the whole lifecycle
- Regulation focuses on products with digital elements
- Classification of products into different classes + exclusion criteria

4 Objectives

1. Improve security throughout the whole lifecycle
2. Ensure coherent framework for cybersecurity
3. Increase transparency of security features of products
4. Enable companies/ consumers to use digital products safely

OTH REGENSBURG

Challenges for manufacturers of IoT products

- How to define requirements for a software that could assists manufacturers with complying with the CRA?
- Additional overhead for manufacturers to proof compliance
- How can compliance be proven to a third party?

OTH REGENSBURG

Presented Solutions

- Research Question: How could a software prototype look like, that assists manufacturers with vulnerability detection to comply with the CRA?

- Idea: Introducing Software that performs compliance checks for IoT devices

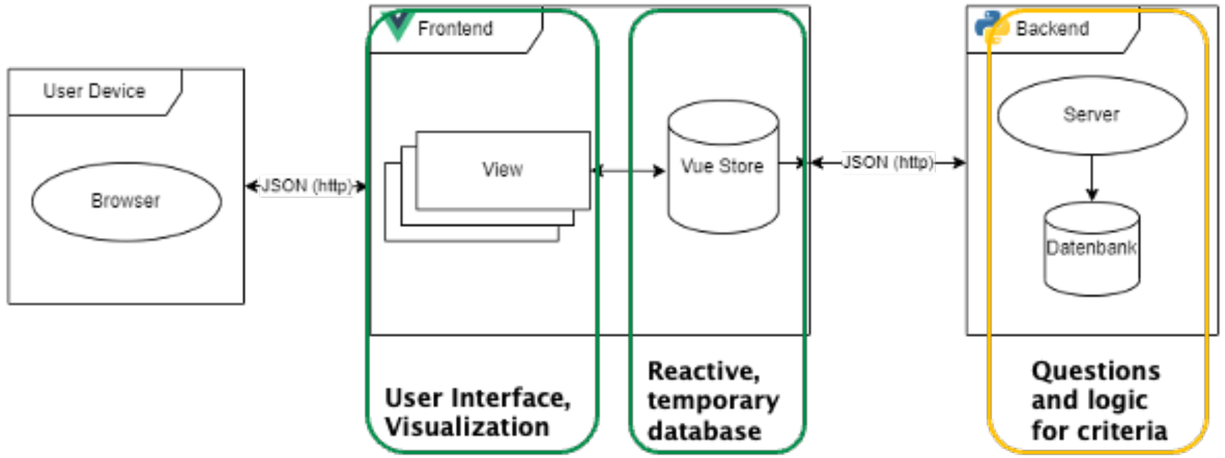
☑ CRA Compliance Checklist

☑ Vulnerability Scanner

OTH REGENSBURG

CRA Compliance Checklist

- A tool to determine the current cyber security standard and to monitor compliance with the requirements of the Cyber Resiliance Act
- Documentation on the current status and the degree of compliance with the regulation
- Tips and information on compliance with the requirements

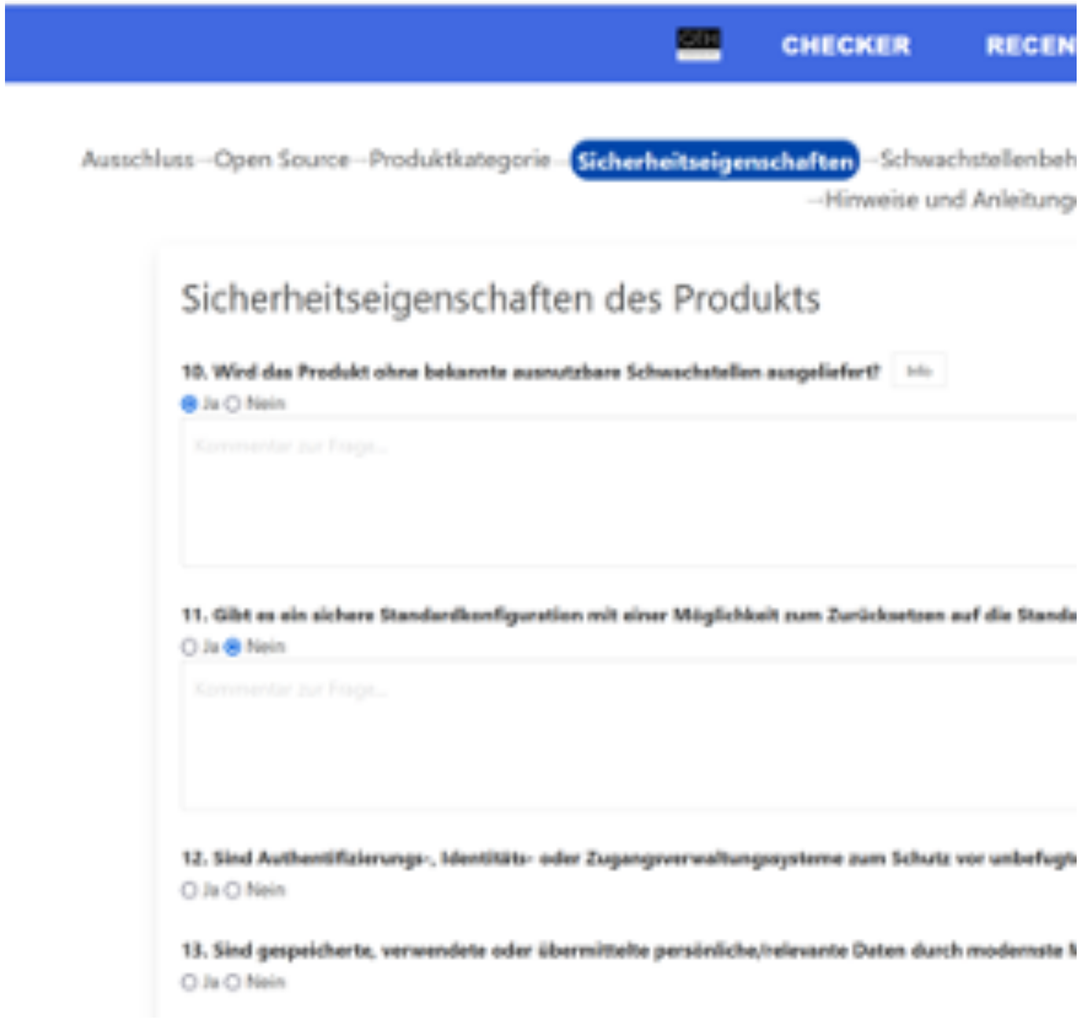


OTH REGENSBURG

CRA Compliance Checklist

e.g. Checklist:

- Is the product delivered without known exploitable weak points?
- Yes / No
- Comments (for documentation)
- Further Information



OTH REGENSBURG

Vulnerability Scanner

Functional Requirements	Non-Functional Requirements
Vulnerability Reporting	**Usability:** Should enable non-developers, like project managers, to track the security status of the project
Software Bills of Materials	**Deployment:** Easy installation procedure, ability to deploy on all platforms
Languages: C, C++ and Python	**Development Process:** Should be integrated into the development process

OTH REGENSBURG

Vulnerability Scanner

Architecture: Microservice Architecture + REST Interface

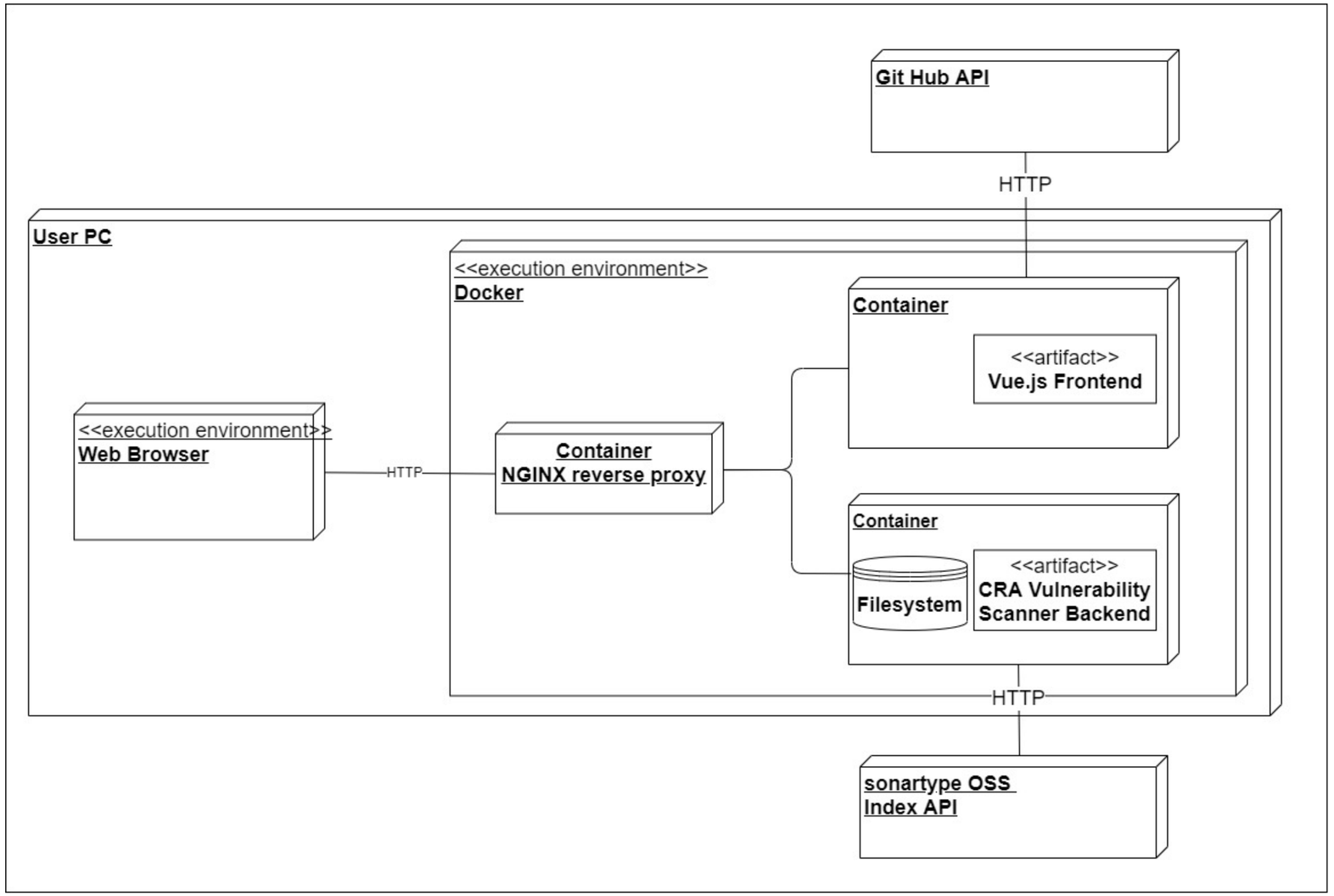
Backend: Python + FAST API

Frontend: Vue.js

Vulnerability Scanning: SBOMs (Syft + OSS Index), Static Application Security Testing (SAST) Tools (e.g Semgrep)

OTH REGENSBURG

Vulnerability Scanner



OTH REGENSBURG

Vulnerability Scanner

Variety of different scanners exit on the market

Scanners
Sonar Qube
Frama-C
CppCheck
Flawfinder
…

Preselection based on criteria

Criteria
Must support C, C++, Python
Open Source
Can handle nested file structures
Can run without library dependencies
Project still maintained

Final Selection
Cppcheck
Flawfinder
Horusec
Semgrep

OTH REGENSBURG

Vulnerability Scanner

Ground Truth Data

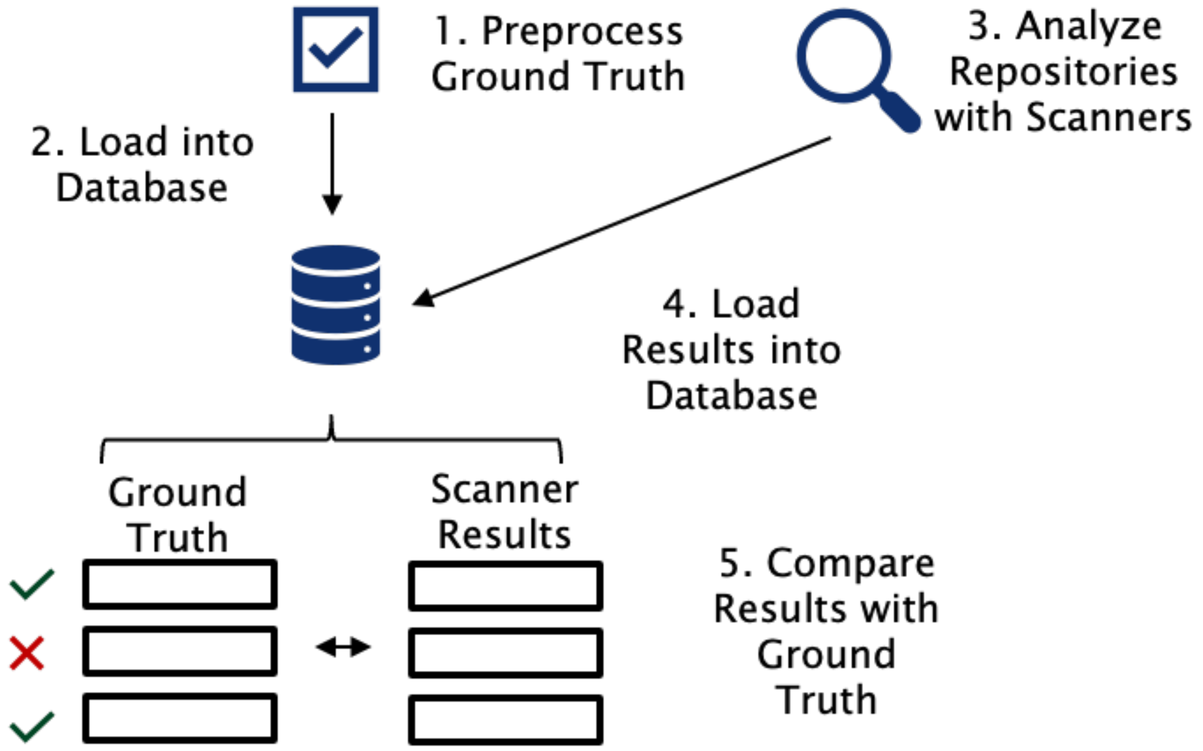
Juliet Test Suite for C++ 1.3.0

- **synthetic** test data
- 64099 Test Cases
- Published by National Institute of Standards and Technology (NIST)

Wireshark 1.8.0

- **real** project
- 127 test cases

Methodology



OTH REGENSBURG

Vulnerability Scanner

Problem

Scanners: have different **output formats**, include different information (e.g Severity, CWE-Mapping)

Assumption: If a hit of scanner matches the exact **file and line** of a flaw in the ground truth data, then this is considered a **true positive**

Evaluation Procedure – Matching Lines

✓ True Positive

Ground Truth		Scanner Result
File x line 12	↔	File x line 12

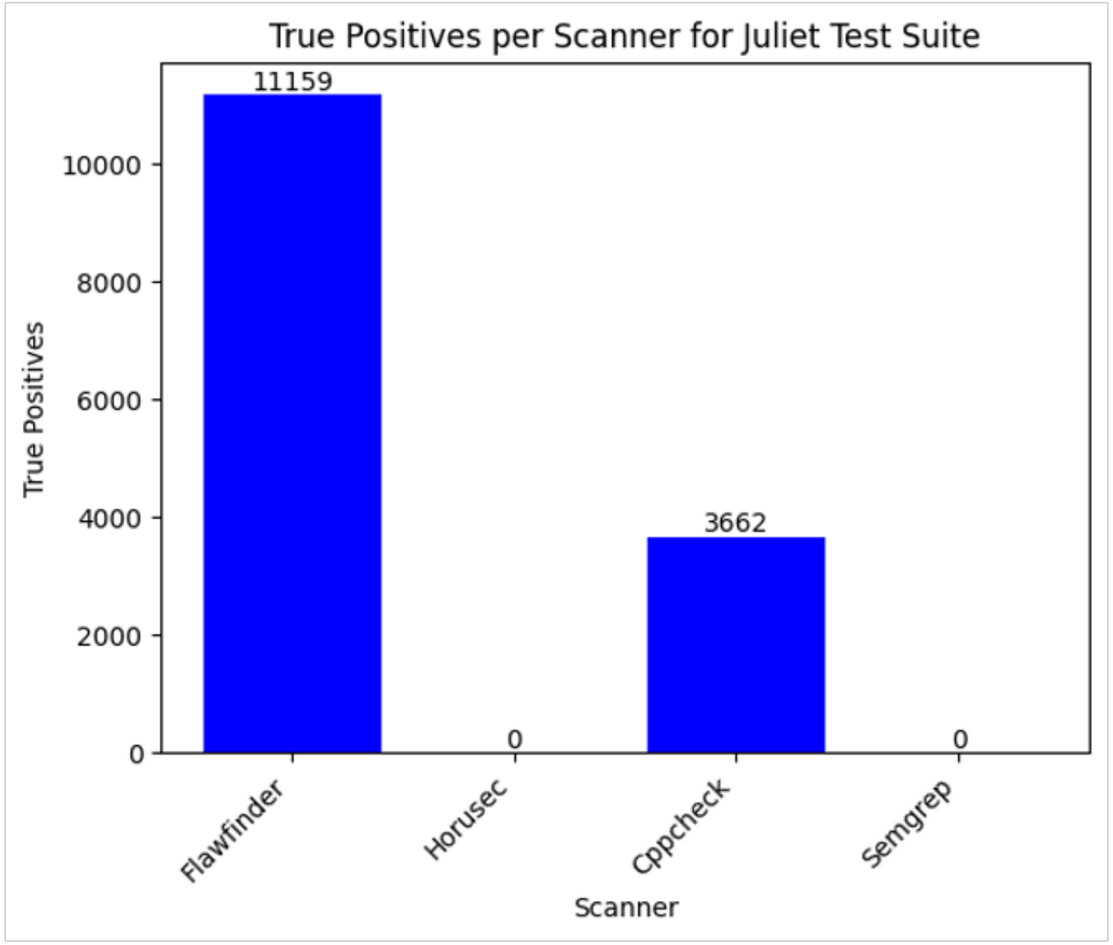
✗ False Positive

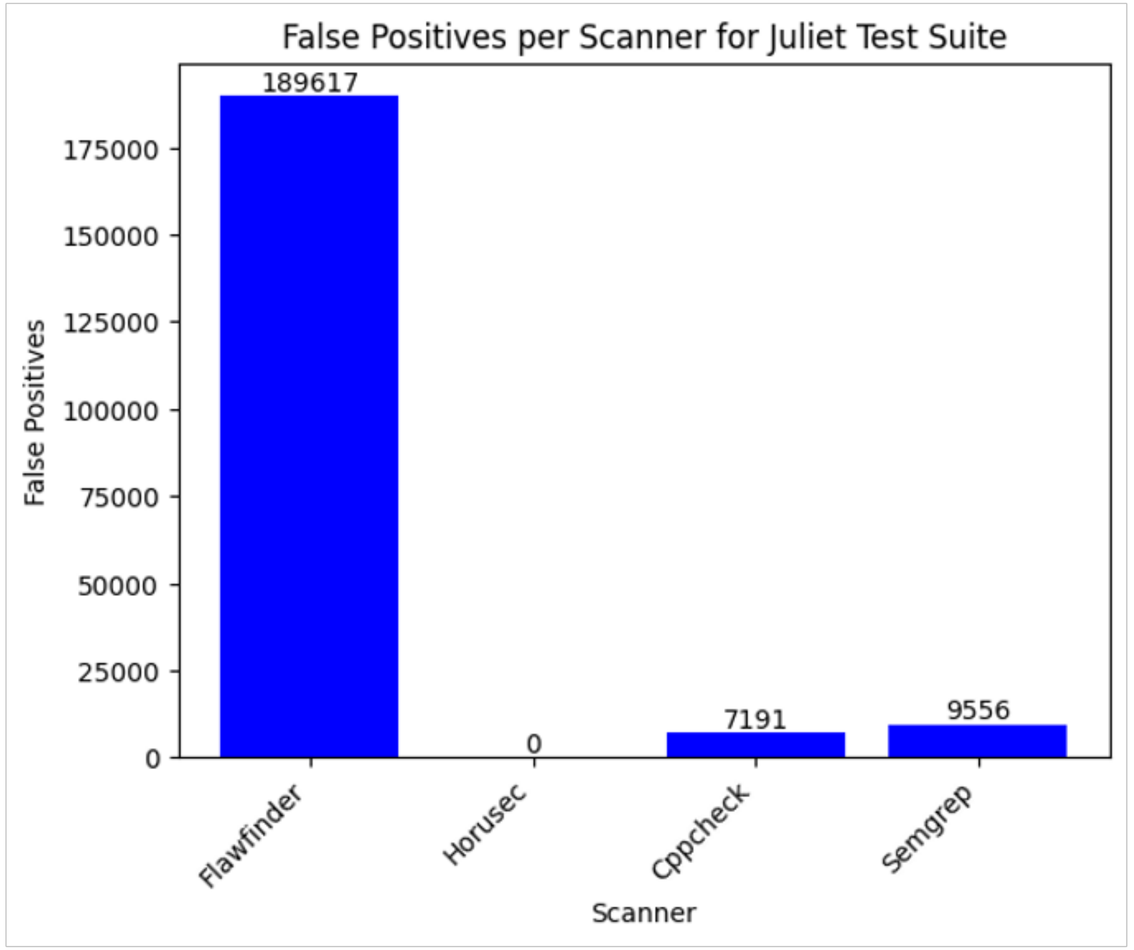
Ground Truth		Scanner Result
File x line 12	↔	File x line 13

OTH REGENSBURG

Vulnerability Scanner

Juliet Test Suite: LoC – 28 394 004, 40 626 Flaws

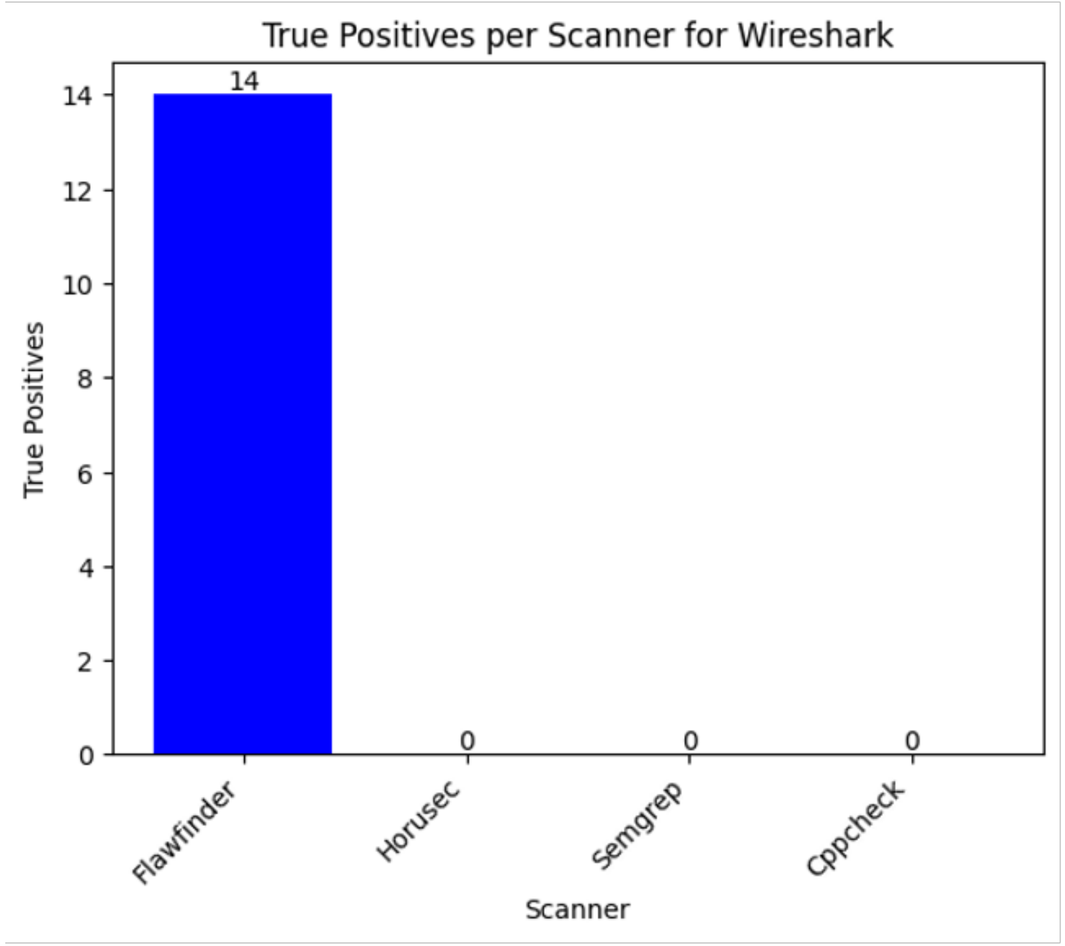


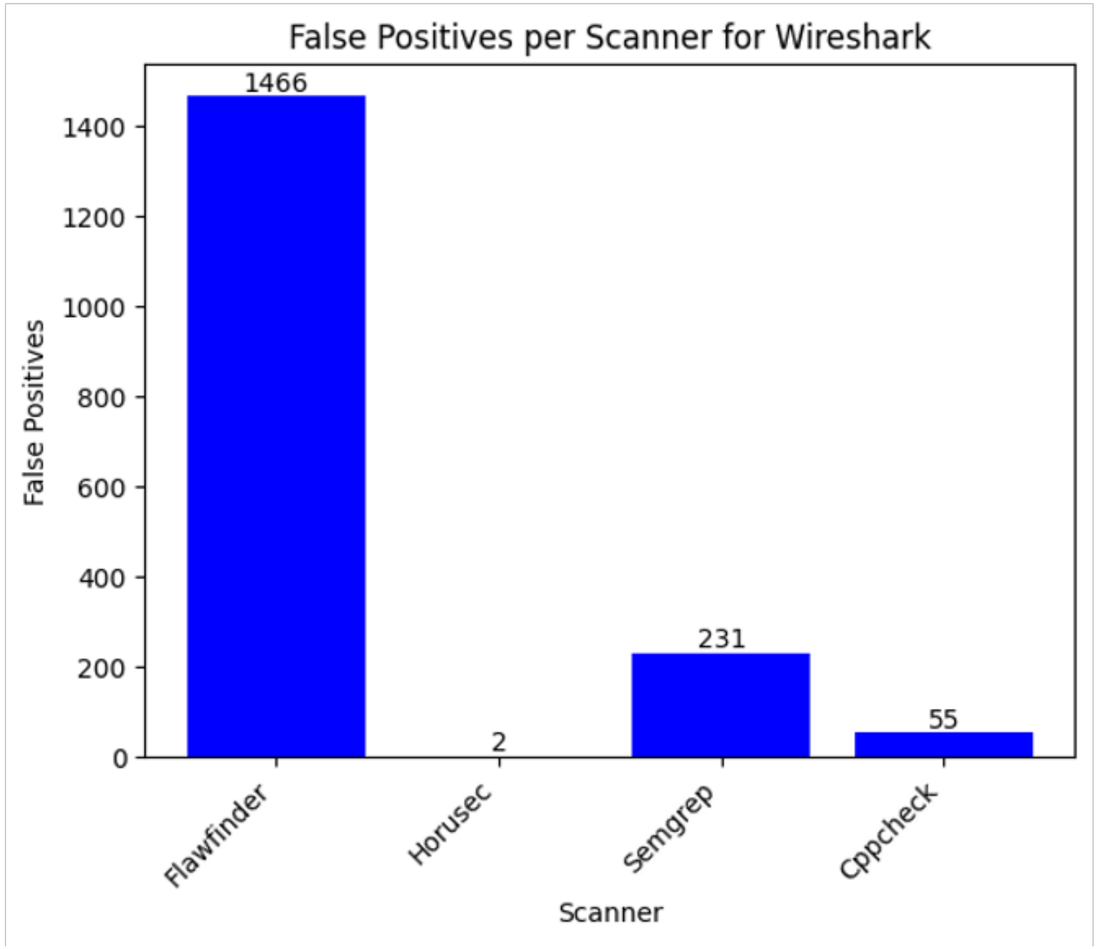


OTH REGENSBURG

Vulnerability Scanner

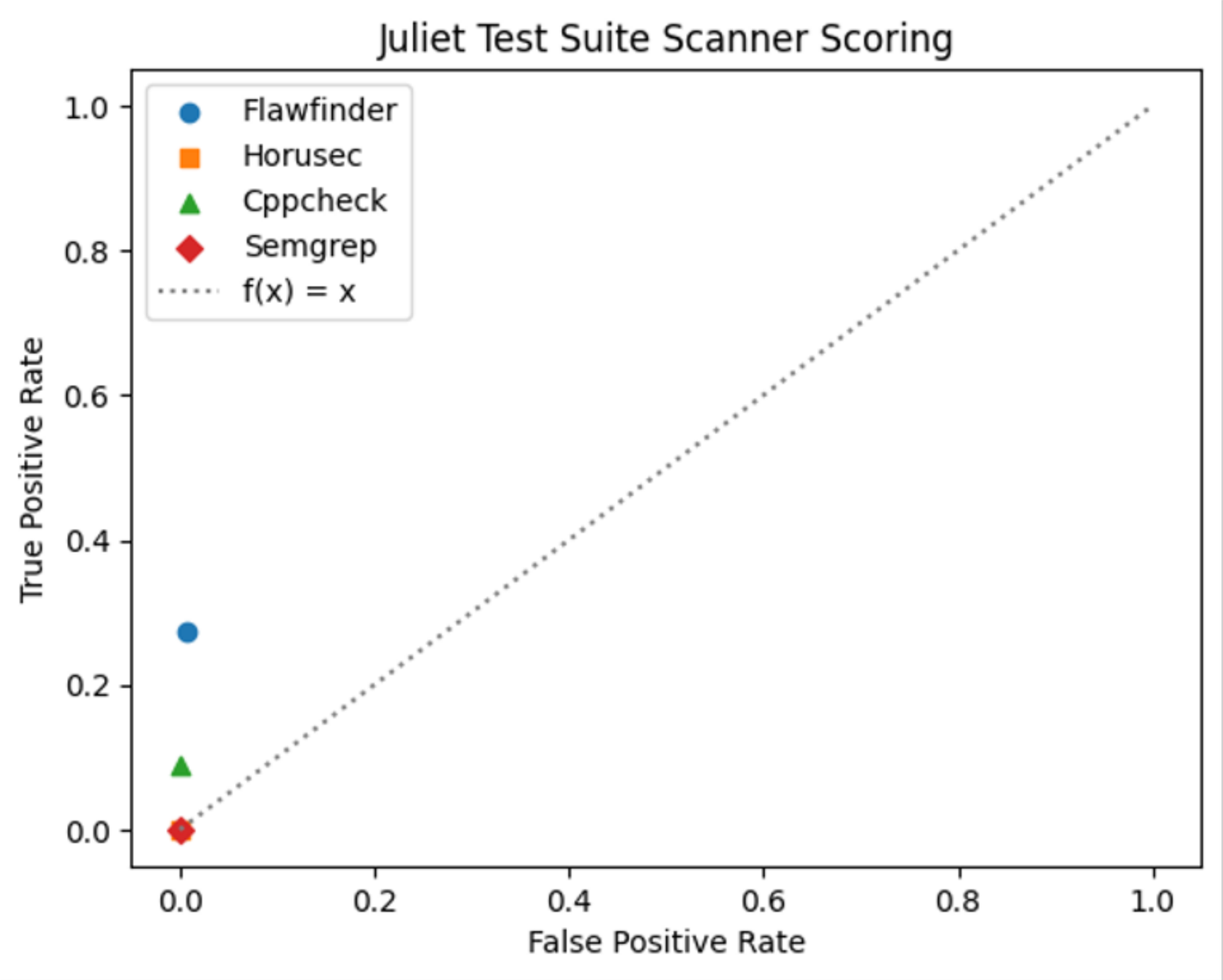
Wireshark: LoC – 1 007 501, 767 Flaws





OTH REGENSBURG

Vulnerability Scanner



Results:
- Flawfinder performed best
- followed by CppCheck

In general:
- SAST Scanners have problems finding most of the vulnerabilities, especially with non-synthetic data

OTH REGENSBURG

Vulnerability Scanner

- SAST Scanners like Semgrep, Flawfinder perform pattern matching
- Quality of the scan depends on variety, number and quality of the patterns
- Certain vulnerabilities cannot be expressed with simple patterns

OTH REGENSBURG

Conclusion / Outlook

Summary:

- ✓ Summarizing the **CRA** and deriving **requirements** for a vulnerability scanning tool
- ✓ Creating a **proof-of-concept** that implements and **visualizes** vulnerabilities
- ✓ Creating a methodology to verify performance of the tool
- ✓ Comparing the performance of multiple SAST scanners

Future Directions:

SAST Tools are very helpful early in the design stage to find vulnerabilities in the code

- Implement new rulesets to detect more severe vulnerabilities
- New approaches: deep learning-based pattern matching

Reverse Engineering:

- often, customers don't have access to the respective code – enable vulnerability detection by reverse engineering the code

OTH REGENSBURG

Conclusion / Outlook

Summary:

- ✓ Summarizing the **CRA** and deriving **requirements** for a vulnerability scanning tool
- ✓ Creating a **proof-of-concept** that implements and **visualizes** vulnerabilities
- ✓ Creating a methodology to verify performance of the tool
- ✓ Comparing the performance of multiple SAST scanners

Future Directions:

SAST Tools are very helpful early in the design stage to find vulnerabilities in the code

- Implement new rulesets to detect more severe vulnerabilities
- New approaches: deep learning-based pattern matching

-> Toolkit for the CRA

Reverse Engineering:

- often, customers don't have access to the respective code – enable vulnerability detection by reverse engineering the code

References

[1] R. Lemos, “Security Guru: Let’s Secure the Net,” 2024, [Online; accessed: 2024-02-27]. URL https://www.zdnet.com/article/security-guru-lets-secure-the-net/

[2] S. Ahmed, M. Carr, M. Nouh, and J. Merritt, “State of the Connected World,” Tech. rep., World Economic Forum, Jan. 2023.

[3] European Commission, “Regulation of the European Parliament and of the Council on horizontal cybersecurity requirements for products with digital elements and amending Regulation (EU) 2019/1020,” 2022, [Online; accessed: 2024-02-27]. URL https://eur- lex.europa.eu/legal- content/ENTXT/?uri=celex: 52022PC0454

[4] C. Skouloudi, A. Malatras, R. Naydenov, and G. Dede, “Guidelines for Securing the Internet of Things,” Tech. rep., ENISA, 2020.

[5] ENISA, “Baseline Security Recommendations for IoT in the Context of Critical Information Infrastructures,” Tech. rep., European Union Agency For Network And Information Security, Nov. 2017.

[6] J. P. Castellanos Ardila, B. Gallina, and F. Ul Muram, “Compliance Checking of Software Processes: A Systematic Literature Review,” Journal of Software: Evolution and Process, 34(5), p. e2440, 2022, ISSN 2047-7481, doi:10.1002/smr.2440.

[7] M. Barati, G. Theodorakopoulos, and O. Rana, “Automating GDPR Compliance Verification for Cloud-hosted Services,” in 2020 Inter- national Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6, Oct. 2020, doi:10.1109/ISNCC49221.2020.9297309.

[8] J. F. Car´ıas, S. Arrizabalaga, L. Labaka, and J. Hernantes, “Cyber Resilience Self-Assessment Tool (CR-SAT) for SMEs,” IEEE Access, 9, pp. 80741–80762, 2021, ISSN 2169-3536, doi:10.1109/ACCESS.2021. 3085530.

[9] RiskOptics, “RZenGRC,” 2024, [Online; accessed: 2024-02-27]. URL https://reciprocity.com/product/zengrc/

[10] cloudsmith, “cloud native artifact management,” 2024, [Online; ac- cessed: 2024-02-27].
URL https://cloudsmith.com/product/cloud-native-artifact-management

[11] F. Lombardi and A. Fanton, “From DevOps to DevSecOps Is Not Enough. CyberDevOps: An Extreme Shifting-Left Architecture to Bring Cybersecurity within Software Security Lifecycle Pipeline,” Software Quality Journal, 31(2), pp. 619–654, Jun. 2023, ISSN 1573-1367, doi: 10.1007/s11219- 023- 09619- 3.

[12] “Vue.Js,” 2024, [Online; accessed: 2024-02-27]. URL https://vuejs.org/

[13] “Git,” 2024, [Online; accessed: 2024-02-27]. URL https://git-scm.com/

[14] “Anchore/Syft,” Anchore, Inc., Jan. 2024.

[15] “Semgrep — Find Bugs and Enforce Code Standards,” 2024, [Online; accessed: 2024-02-27]. URL https://semgrep.dev/

[16] “Flawfinder Home Page,” 2024, [Online; accessed: 2024-02-27]. URL https://dwheeler.com/flawfinder/

[17] “Cppcheck - A Tool for Static C/C++ Code Analysis,” 2024, [Online; accessed: 2024-02-27]. URL https://cppcheck.sourceforge.io/

[18] “Horusec,” 2024, [Online; accessed: 2024-02-27]. URL https://horusec.io/site/

[19] “Docker: Accelerated Container Application Development,” May 2022, [Online; accessed: 2024-02-27]. URL https://www.docker.com/

[20] “GitHub:Let’sBuildfromHere,”2024,[Online;accessed:2024-02-27]. URL https://github.com/

[21] S. Inc, “Sonatype OSS Index,” 2024, [Online; accessed: 2024-02-27]. URL https://ossindex.sonatype.org/

[22] C.Gentsch,“EvaluationofOpenSourceStaticAnalysisSecurityTesting (SAST) Tools for C,” Technical Report DLR-IB-DW-JE-2020-16, DLR German Aerospace Center, Jan. 2020.

[23] National Institute for Standards and Technology, “Juliet C/C++ 1.3 - NIST Software Assurance Reference Dataset,” 2017, [Online; accessed: 2024-02-27].URL https://samate.nist.gov/SARD/test-suites/112

[24] National Institute for Standards and Technology, “Wireshark 1.8.0 - NIST Software Assurance Reference Dataset,” 2014, [Online; accessed: 2024-02-27].URL https://samate.nist.gov/SARD/test-suites/94

[25] D. A. Wheeler, “Flawfinder/Flawfinder.Py at Master · David- a-Wheeler/Flawfinder · GitHub,” https://github.com/david-a- wheeler/flawfinder/blob/master/flawfinder.py.

[26] “Semgrep-Rules/c/Lang/Security at Develop · Semgrep/Semgrep- Rules · GitHub,” https://github.com/semgrep/semgrep- rules/tree/develop/c/lang/security.