Automated Language-Agnostic Multi-Metric Security Evaluation for Transpilation Workflows

www.thalesgroup.com

THALES
Building a future we can all trust

VUB VRIJE UNIVERSITEIT BRUSSEL

ROYAL HIGHER INSTITUTE for DEFENCE

RMA Brussels
The only Belgian military university

Table of contents

Why Security Evaluation Matters

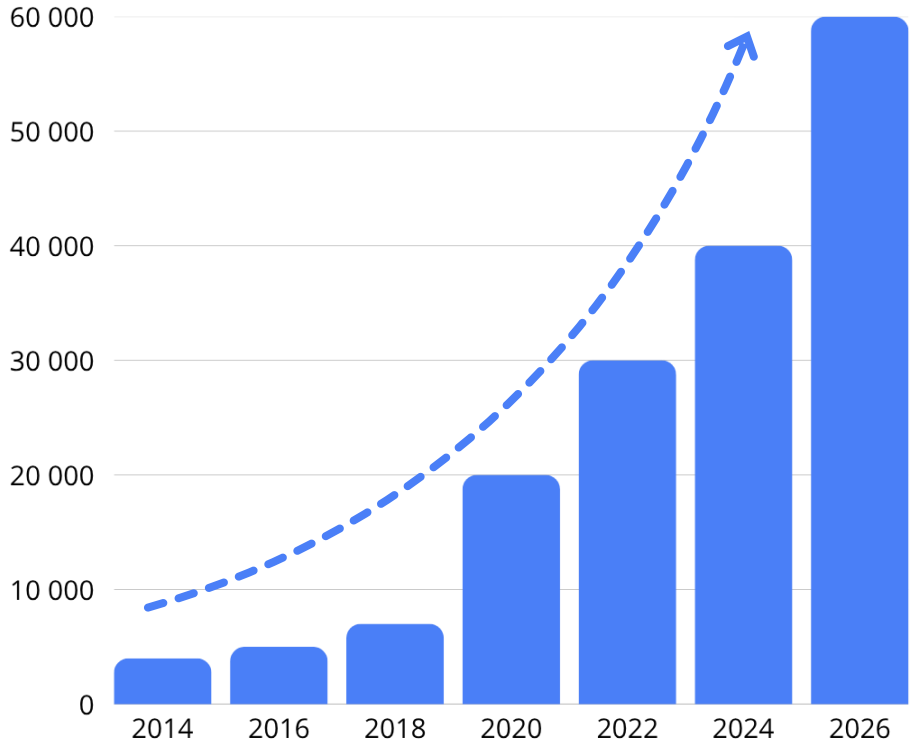
Memory vulnerabilities
Remain a major source of security flaws

Legacy systems
Are still largely written in unsafe languages.

Increasing attacks
Memory-safety bugs are a growing threat.

Memory-safe migration
Industry is moving toward memory-safe languages.

Quantity of CVE discovered every two years between 2014 and 2026



Security claims require measurable evidence

Why Comparison Is Difficult

Different Analyzers
Each tool uses different rules, configurations and assumptions.

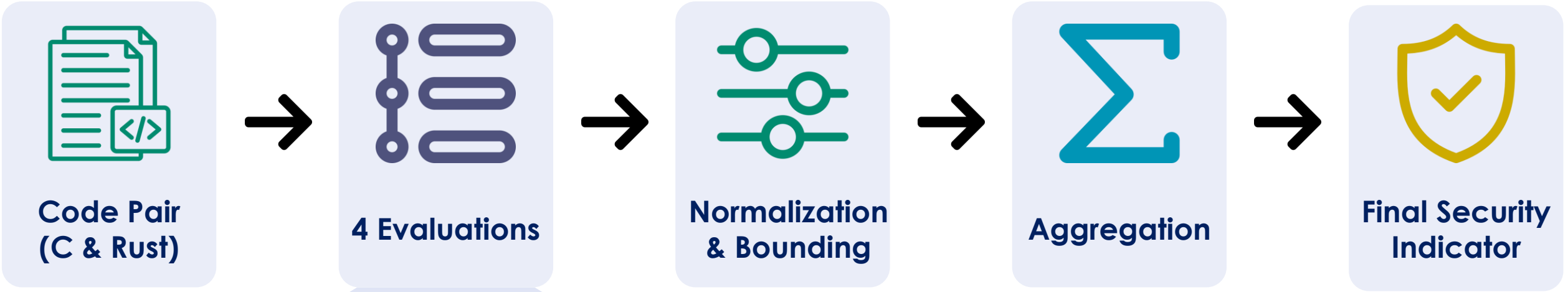
Different Scales
Raw outputs and severities are not comparable

Different Logic
All languages are different. Different code structure,…

Different Visibility
Not all vulnerabilities are detectable in the same way.

Raw security outputs cannot be directly compared.

Overall Evaluation Pipeline



Evaluation method:

Static Analysis
Broad Code Coverage

Memory Vulnerability
Focus on Memory Safety

Dynamic
Runtime Evidence & ML-Triage

Delta
Measure Improvement

Normalization score:

RAW STATIC FINDINGS

More functions → More detections

Project A

120
Vulnerabilities Found

Project B

600
Vulnerabiliies Found

AFTER NORMALIZATION

$$\frac{120\ findings}{1{,}000\ functions} = 0.12$$

$$\frac{600\ findings}{10{,}000\ functions} = 0.06$$

IMPACT OF DENSITY

RAW COUNTS ARE MISLEADING

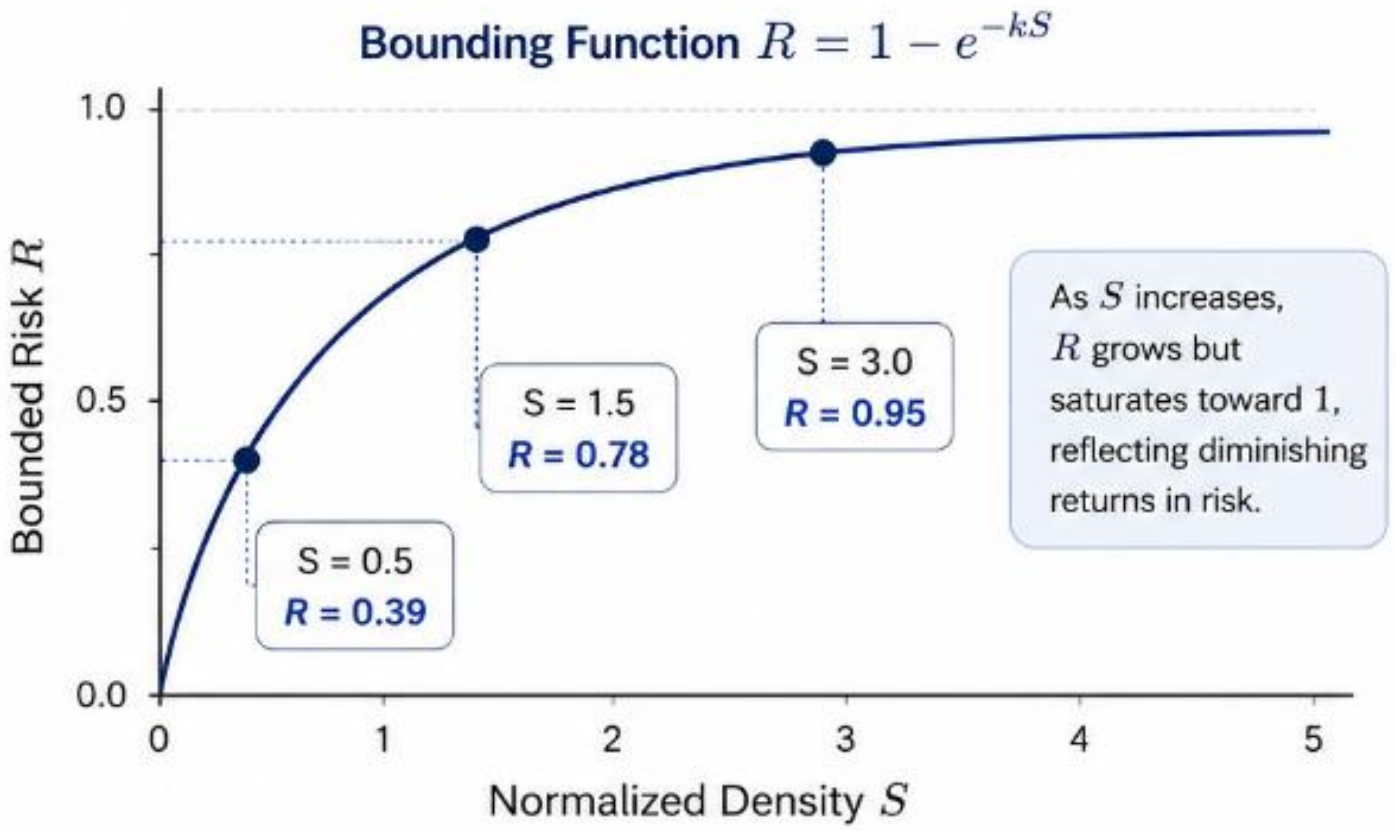
Project B looks worse
(600 > 120)

NORMALIZATION REVEALS THE TRUE PICTURE

Project A is actually riskier
(0.12 > 0.06)

Why Bounding Is Necessary

- **Common Scale [0,1]**
- **Interpretable risk indicator**
- **Limits the impact of extremes**
- **Preserves ranking (monotonic)**



$$R = 1 - e^{-kS}$$

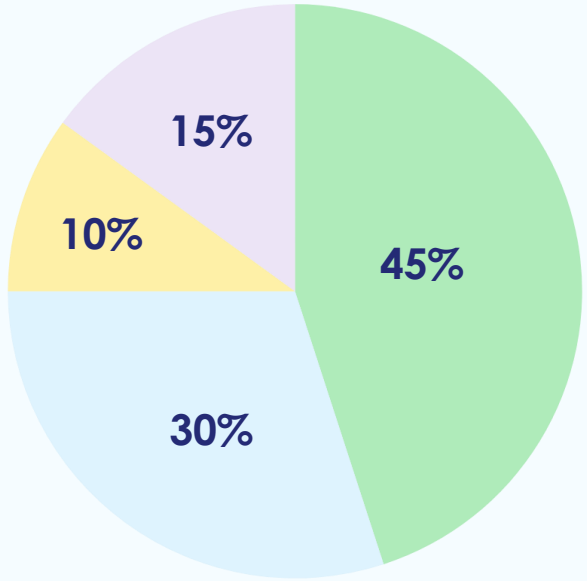
S: normalized density
k: bounding parameter

Meaningful metric

Normalized Sub-scores

- Static Score
- Dynamic Score
- Memory Score
- Delta Score

Weighted Aggregation



Final Security Indicator (SI)

A single, interpretable metric for overall security

Experimental Setup

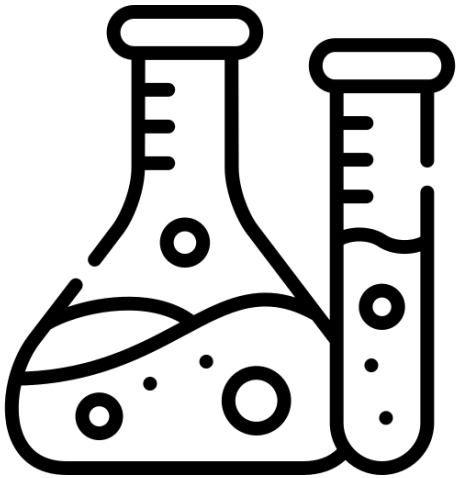
Dataset
CWE-based vulnerable programs

Languages
C (Original) and Rust (Transpiled)

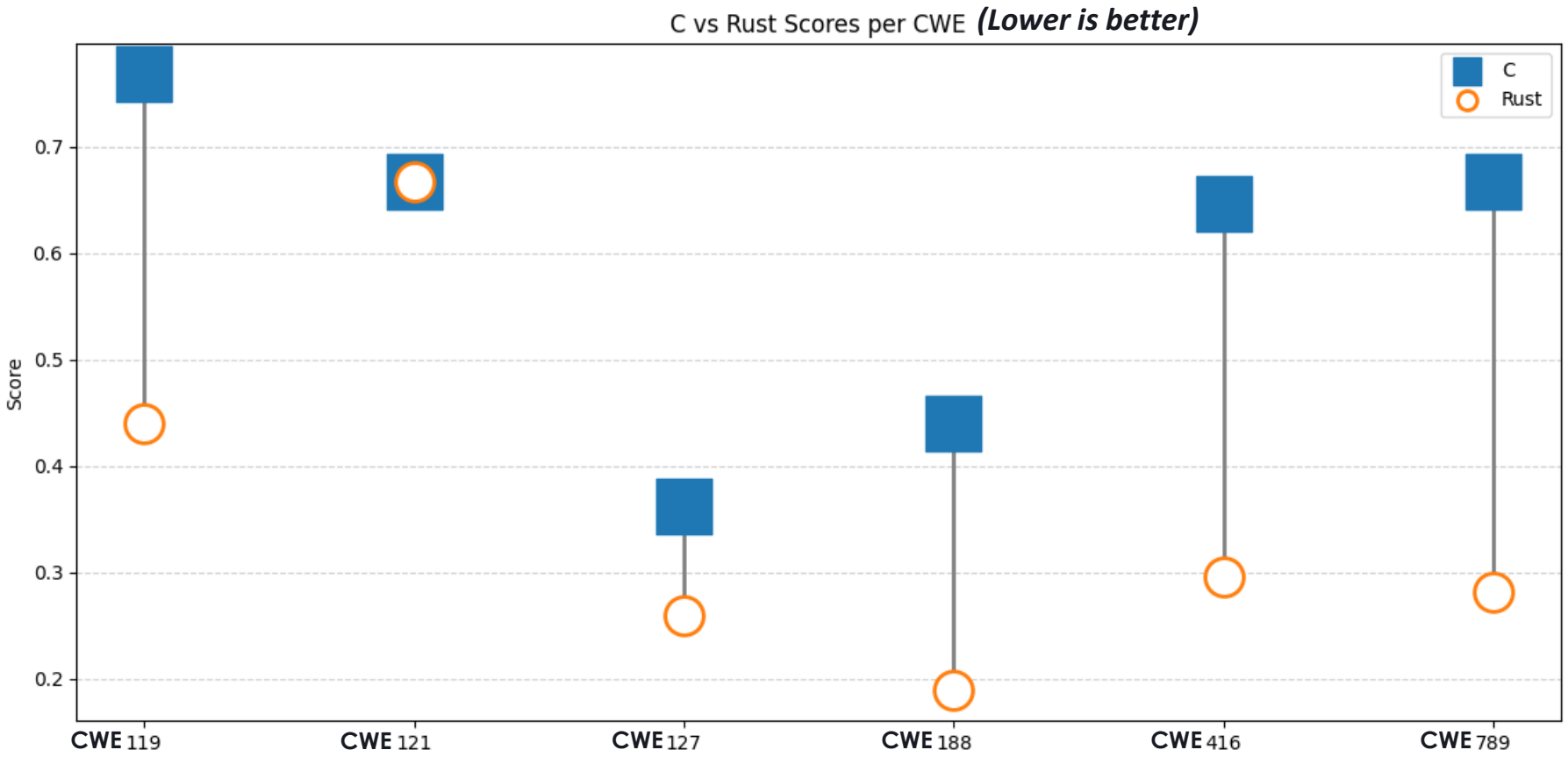
Static Tools
SonarQube,Rudra, Clippy, …

Dynamic Tools
AFL++, Asan, UBSan, …

Environment
Ubuntu 24.04.2 LTS



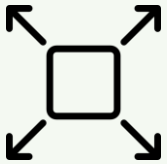
Result Overview (Normalized Scores)



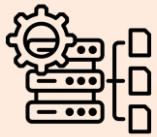
Rust provides better score than C

Future Work

Evaluate on larger, real-world industrial codebase



Extend to more CWE categories and vulnerability types



Integrate the framework into CI/CD pipelines

Integrate ML to generate fuzzing seeds

Improve ML-assisted triage to reduce false positives