SoECM: Survey of Enhanced Coverage Metrics

Andrew D. Barlow, Jason Garcia Solorzano, Charles Nicholas, Jeffrey Reginald, Carolyn Seaman, Chinyere Sloley
UMBC, Baltimore, MD

Steve Bailey, Don Goff, Dan Wolf
Cyber Pack Ventures, Inc.

Dr. Paul Black
NIST

The Challenge Problem on ***Static Analysis Coverage***

Abstract

Background:

- Static analyzers are software tools that provide detailed indications of potential software vulnerabilities (either malicious or unintentional) that exist in a codebase under test
- These analyzers help developers and users to identify and mitigate weaknesses in the code.

The Problem:

- Most static analyzers do not give an indication of what was and was not actually evaluated.
- If few weaknesses are reported, does that mean there aren't many weaknesses or did the tool not analyze some areas of the code?
- To instill confidence in the software being analyzed, we need a mechanism to report the actual coverage of static analysis tools, including what code segments or basic blocks or modules were examined.
- Such information would facilitate a more thorough risk assessment when incorporating the analyzed software as part of the supply chain.

Research goals:

- Evaluate a representative sample of static analyzers to determine the actual code coverage they achieve
- Develop a simple prototype to demonstrate how coverage information could be captured and reported by a static analyzer

Approach:

For open source analyzers:

- installing the analyzer
- exploring its functions
- reviewing the open-source code to understand coverage
- instrumenting the analyzer to collect coverage information
- running the analyzer on the sample code
- recording vulnerabilities found
- measuring code coverage.

For commercial analyzers:

- acquiring and installing the analyzer
- exploring its functions
- running the tool on sample code with known vulnerabilities
- recording the vulnerabilities found
- estimating coverage based on vulnerabilities found

Prototype:

- instrumenting CPPCheck (an open source static analyzer) to report each file it examines
- capturing the examined file output and comparing it to total number of files in the sample code
- displaying the coverage percentage after the run

Results:

- 10 static analyzers were evaluated (6 open source)
- All appeared to cover 100% of the sample code being analyzed, although results were not clear for commercial tools
- Analyzers varied in terms of how many of the known vulnerabilities they found, based on:
  - Which vulnerability types were covered
  - What rules were used to detect different vulnerability types
  - What level of confidence the analyzer used to determine if a potential vulnerability was reportable
- Instrumenting an open source analyzer to report coverage information (as we did in our prototype) is a reasonable way to get such information to the user

Screen shot of prototype output:

```
Checked 146564 configurations.
Checked 11414/11422 files.
Coverage: 99.93%
```

- Screen shot of prototype output: