The Impact of Code Complexity on Static Analysis Results

James Walden, Adam Messer

Northern Kentucky University

July 29, 2008

Outline

1. Research Objective
2. Test Cases
3. Metrics
4. Results
5. Analysis

Research Objective

Goal: Identify effects of code complexity on static analysis results.

Precision vs Scalability Tradeoff:

Increasing precision takes more time, decreasing size of code that can be analyzed in an acceptable amount of time.

Selected Prior Work:

- [Zitser, Lippmann, Leek 2004]
- [Kratkiewicz, Lippmann 2005]
- SAMATE

Study Needs

- A static analysis tool
  - Fortify Source Code Analyzer 4.5.0
- A vulnerability type that is reliably identified
  - Format string
- Metrics
  - Static analysis quality
  - Code complexity
- Test cases
  - Vulnerable and fixed source code samples

Metrics

- Static Analysis Metrics
  - Detection rate
  - False positive rate
- Code Metrics
  - Source Lines of Code (SLOC)
  - Cyclomatic Complexity

Test Cases

35 format string vulnerabilities
- Selected randomly from NVD.
- Open source C/C++ code that compiles on Linux.
- Each case has two versions of the code
  - One version has a format string vulnerability.
  - Other version is same program with vulnerability fixed.

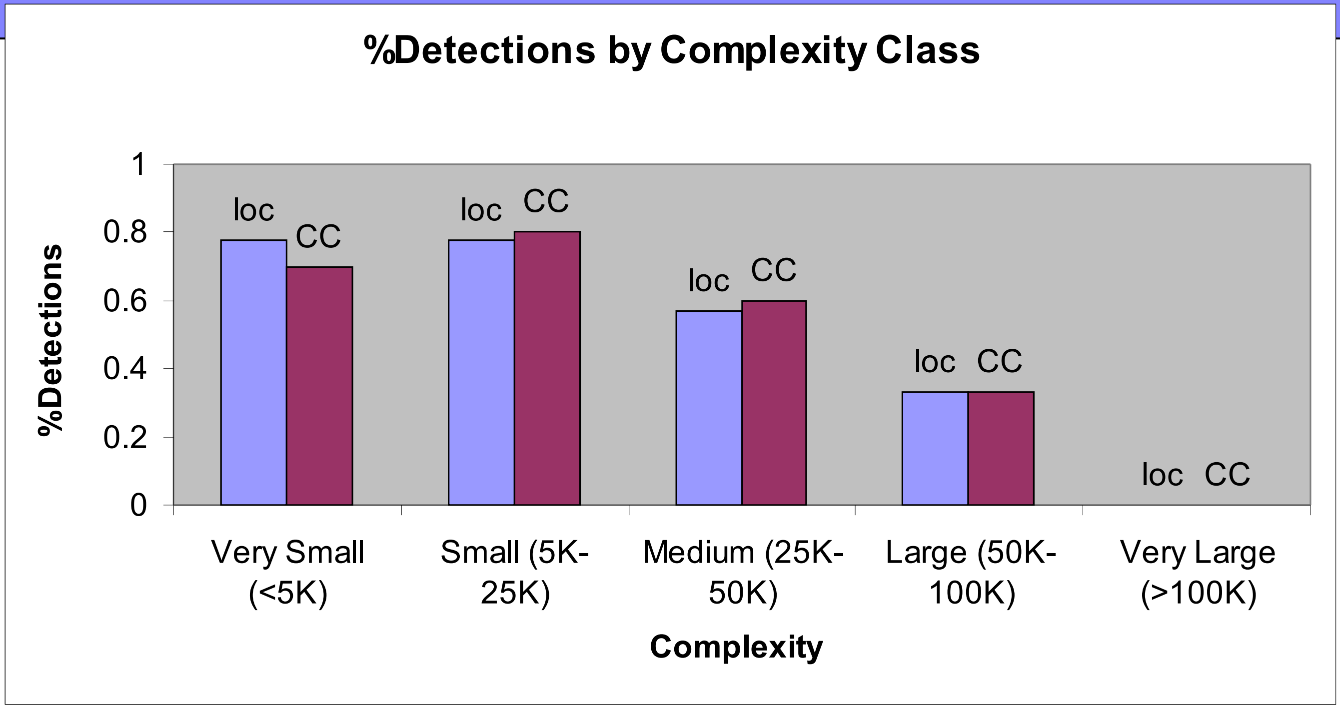
Examples

- wu-ftpd
- screen
- stunnel
- gpg
- hylafax
- exim
- dhcpd
- squid
- Kerberos 5
- cdrtools
- gnats
- cvs
- socat
- ethereal
- openvpn

Results

- Detections
  - 22 of 35 (63%) flaws detected by SCA 4.5.
- Detections by Complexity
  - Divided samples into 5 complexity bins.
  - No significant difference between SLOC and CC.
- Discrimination:
  - Measure of how often analyzer passes fixed test cases when it also passes vulnerable case.
  - Results almost identical to detection results since
  - Only one false positive from 35 fixed samples.

Detections by Complexity Class



Class	Lines of Code	Samples	Cyclomatic	Samples
Very Small	< 5000	9	< 1000	10
Small	5000 – 25,000	9	1000 – 5000	10
Medium	25,000 – 50,000	7	5000 – 10,000	5
Large	50,000 – 100,000	6	10,000 – 25,000	6
Very Large	> 100,000	4	> 25,000	4

Why do static analysis detection rates decrease with complexity?

Hypothesis 1: Tool designers make tradeoffs between precision and scalability, reducing the depth of analysis to handle larger programs in a reasonable amount of time.

Hypothesis 2: Software changes as it grows more complex, with increasing use of custom libraries such as the Apache Portable Runtime, which are not included in the rulesets of tools.

Problem: How do we measure the relative effect of each hypothesis? Are there alternative hypotheses?

Characteristics of Large Software

1. More complex control + data flow.
2. Participation of multiple developers.
3. Use of a broader set of language features.
4. Increased use of custom libraries.

Future Work

- How do static analysis results change with time? What happens after we remove all of the bugs that can be detected?
- How does code size affect the number of vulnerabilities in a program over time? How does churn affect this?