Carnegie Mellon University

Static and Dynamic Analysis

17-313, Foundations of Software Engineering, Fall 2022

S3D
Software and Societal Systems Department

Administrivia

- HW4: Ethical Reflection due Thursday (Nov 10)

Learning Goals

- Gain an understanding of the relative strengths and weaknesses of static and dynamic analysis
- Examine several popular analysis tools and understand their use cases
- Understand how analysis tools are used in large open source software

Activity: Analyze the Python program statically

```python
def n2s(n: int, b: int):
  if n <= 0: return '0'
  r = ''
  while n > 0:
    u = n % b
    if u >= 10:
      u = chr(ord('A') + u-10)
    n = n // b
    r = str(u) + r
  return r
```

1. What are the set of data types taken by variable `u` at any point in the program?
2. Can the variable `u` be a negative number?
3. Will this function always return a value?
4. Can there ever be a division by zero?
5. Will the returned value ever contain a minus sign ‘-’?

What static analysis can and cannot do

- Type-checking is well established
  - Set of data types taken by variables at any point
  - Can be used to prevent type errors (e.g. Java) or warn about potential type errors (e.g. Python)

- Checking for problematic patterns in syntax is easy and fast
  - Is there a comparison of two Java strings using `==`?
  - Is there an array access `a[i]` without an enclosing bounds check for `i`?

- Reasoning about termination is impossible in general
  - Halting problem

- Reasoning about exact values is hard, but conservative analysis via abstraction is possible
  - Is the bounds check before `a[i]` guaranteeing that `I` is within bounds?
  - Can the divisor ever take on a zero value?
  - Could the result of a function call be `42`?
  - Will this multi-threaded program give me a deterministic result?
  - Be prepared for “MAYBE”

- Verifying some advanced properties is possible but expensive
  - CI-based static analysis usually over-approximates conservatively

The Bad News: Rice’s Theorem

Every static analysis is necessarily incomplete, unsound, undecidable, or a combination thereof

> “Any nontrivial property about the language recognized by a Turing machine is undecidable.”

Henry Gordon Rice, 1953

Static Analysis is well suited to detecting certain defects

- **Security:** Buffer overruns, improperly validated input…
- **Memory safety:** Null dereference, uninitialized data…
- **Resource leaks:** Memory, OS resources…
- **API Protocols:** Device drivers; real time libraries; GUI frameworks
- **Exceptions:** Arithmetic/library/user-defined
- **Encapsulation:**
  - Accessing internal data, calling private functions…
- **Data races:**
  - Two threads access the same data without synchronization

Activity: Analyze the Python program dynamically

```python
def n2s(n: int, b: int):
  if n <= 0: return '0'
  r = ''
  while n > 0:
    u = n % b
    if u >= 10:
      u = chr(ord('A') + u-10)
    n = n // b
    r = str(u) + r
  return r

print(n2s(12, 10))
```

1. What are the set of data types taken by variable `u` at any point in the program?
2. Did the variable `u` ever contain a negative number?
3. For how many iterations did the while loop execute?
4. Was there ever be a division by zero?
5. Did the returned value ever contain a minus sign ‘-’?

Dynamic analysis reasons about program executions

- Tells you properties of the program that were definitely observed
  - Code coverage
  - Performance profiling
  - Type profiling
  - Testing
- In practice, implemented by program *instrumentation*
  - Think “Automated logging”
  - Slows down execution speed by a small amount

Static Analysis	Dynamic Analysis
Requires only source code	Requires successful build + test inputs
Conservatively reasons about all possible inputs and program paths	Observes individual executions
Reported warnings may contain false positives	Reported problems are real, as observed by a witness input
Can report all warnings of a particular class of problems	Can only report problems that are seen. Highly dependent on test inputs. Subject to false negatives
Advanced techniques like verification can prove certain complex properties, but rarely run in CI due to cost	Advanced techniques like symbolic execution can prove certain complex properties, but rarely run in CI due to cost

Static Analysis Tools

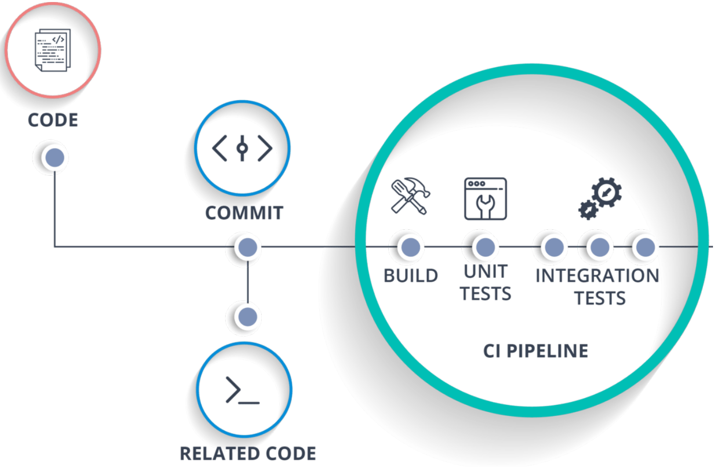
Tools for Static Analysis

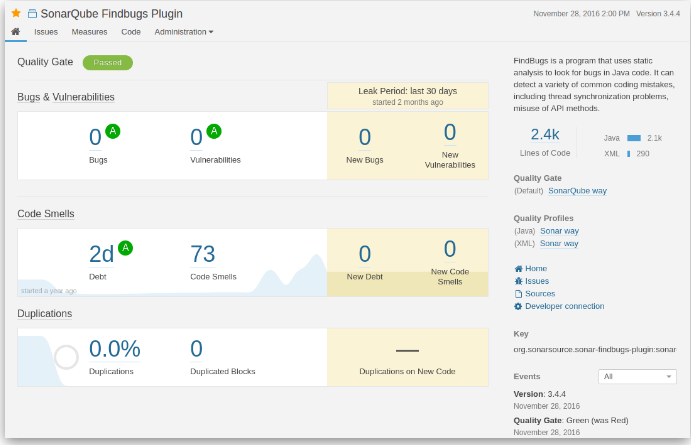
CSi
UNIVERSITY OF MARYLAND
18 56
TM
FindBugs
because it's easy
: my[py]
snyk
sonarqube
coverity®
squale

Static analysis can be applied to all attributes

- Find bugs
- Refactor code
- Keep your code stylish!
- Identify code smells
- Measure quality
- Find usability and accessibility issues
- Identify bottlenecks and improve performance

Static analysis is a key part of continuous integration





SonarQube Findbugs Plugin

November 28, 2016 2:00 PM Version 3.4.4

Quality Gate Passed

Bugs & Vulnerabilities

Leak Period: last 30 days
started 2 months ago

0 A Bugs

0 A Vulnerabilities

0 New Bugs

0 New Vulnerabilities

Code Smells

2d A Debt

73 Code Smells

started a year ago

0 New Debt

0 New Code Smells

Duplications

0.0% Duplications

0 Duplicated Blocks

— Duplications on New Code

FindBugs is a program that uses static analysis to look for bugs in Java code. It can detect a variety of common coding mistakes, including thread synchronization problems, misuse of API methods.

2.4k Lines of Code

Java 2.1k
XML 290

Quality Gate
(Default) SonarQube way

Quality Profiles
(Java) Sonar way
(XML) Sonar way

Home
Issues
Sources
Developer connection

Key
org.sonarsource.sonar-findbugs-plugin:sonar

Events All

Version: 3.4.4
November 28, 2016

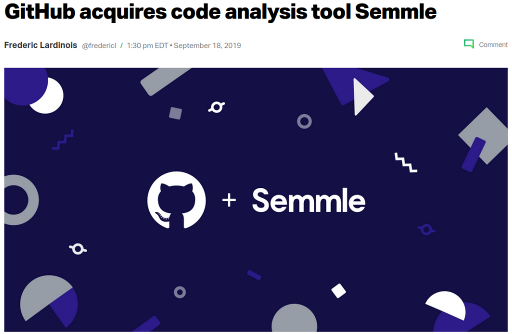
Quality Gate: Green (was Red)
November 28, 2016

Travis CI

GitHub Actions

sonarqube

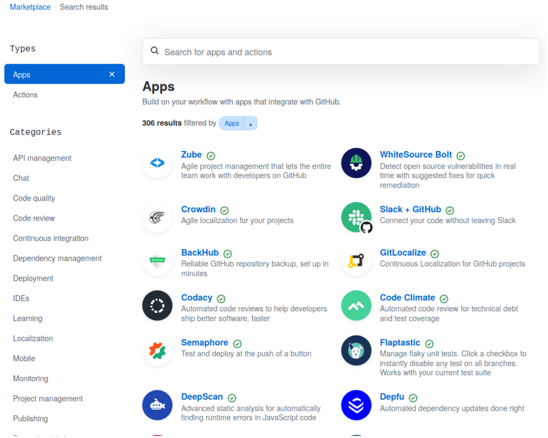
Static analysis is a growing industry



GitHub acquires code analysis tool Semmle

Frederic Lardinois @fredericl / 1:30 pm EDT • September 18, 2019

Comment



Marketplace / Search results

Types

- Apps ✕
- Actions

Categories

- API management
- Chat
- Code quality
- Code review
- Continuous integration
- Dependency management
- Deployment
- IDEs
- Learning
- Localization
- Mobile
- Monitoring
- Project management
- Publishing

Search for apps and actions

Apps

Build on your workflow with apps that integrate with GitHub.

306 results filtered by Apps ×

Zube ✓
Agile project management that lets the entire team work with developers on GitHub

WhiteSource Bolt ✓
Detect open source vulnerabilities in real time with suggested fixes for quick remediation

Crowdin ✓
Agile localization for your projects

Slack + GitHub ✓
Connect your code without leaving Slack

BackHub ✓
Reliable GitHub repository backup, set up in minutes

GitLocalize ✓
Continuous Localization for GitHub projects

Codacy ✓
Automated code reviews to help developers ship better software, faster

Code Climate ✓
Automated code review for technical debt and test coverage

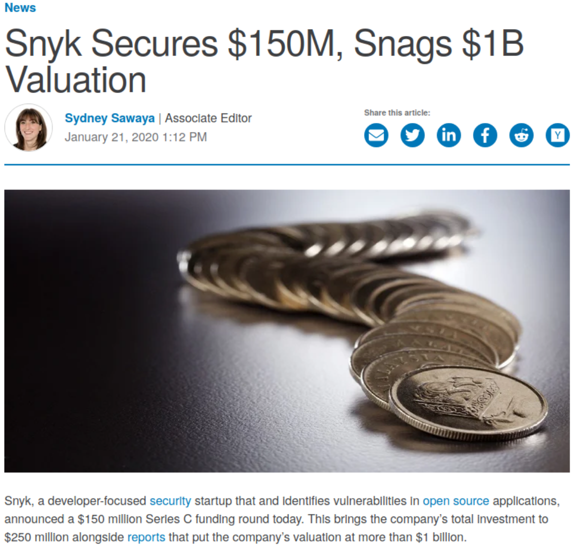
Semaphore ✓
Test and deploy at the push of a button

Flaptastic ✓
Manage flaky unit tests. Click a checkbox to instantly disable any test on all branches. Works with your current test suite

DeepScan ✓
Advanced static analysis for automatically finding runtime errors in JavaScript code

Depfu ✓
Automated dependency updates done right

GitHub



News

Snyk Secures $150M, Snags $1B Valuation

Sydney Sawaya | Associate Editor
January 21, 2020 1:12 PM

Share this article:

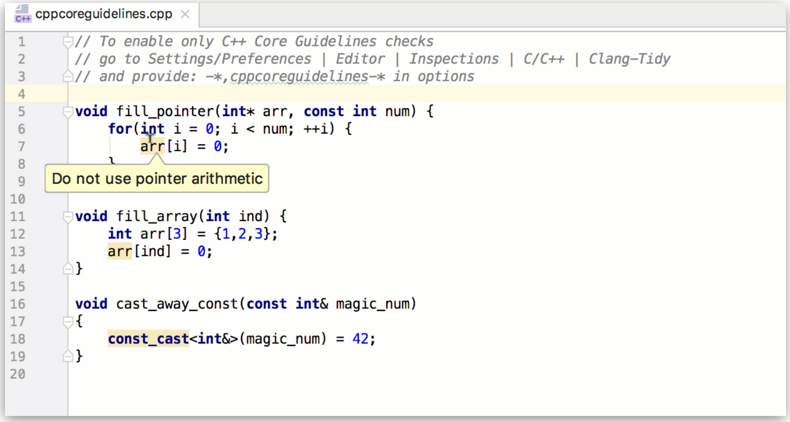
Snyk, a developer-focused security startup that and identifies vulnerabilities in open source applications, announced a $150 million Series C funding round today. This brings the company's total investment to $250 million alongside reports that put the company's valuation at more than $1 billion.

snyk

S3D

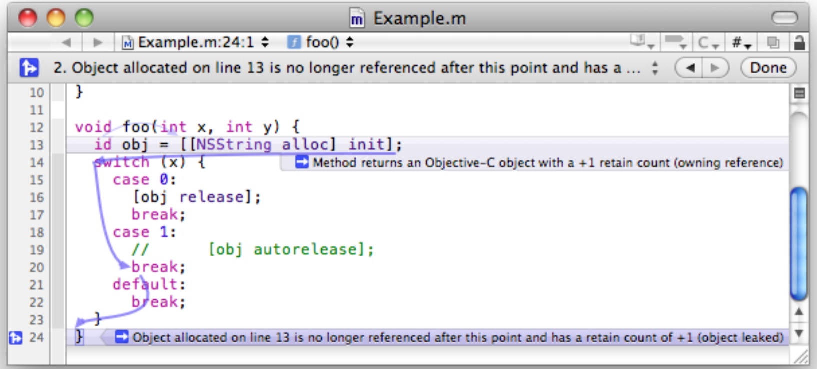
https://www.sdxcentral.com/articles/news/snyk-secures-150m-snags-1b-valuation/2020/01/
https://techcrunch.com/2019/09/18/github-acquires-code-analysis-tool-semmle/
https://github.com/marketplace

Static analysis is also integrated into IDEs



```cpp
// To enable only C++ Core Guidelines checks
// go to Settings/Preferences | Editor | Inspections | C/C++ | Clang-Tidy
// and provide: -*,cppcoreguidelines-* in options

void fill_pointer(int* arr, const int num) {
    for(int i = 0; i < num; ++i) {
        arr[i] = 0;
    }

void fill_array(int ind) {
    int arr[3] = {1,2,3};
    arr[ind] = 0;
}

void cast_away_const(const int& magic_num)
{
    const_cast<int&>(magic_num) = 42;
}
```

Do not use pointer arithmetic



2. Object allocated on line 13 is no longer referenced after this point and has a ...

```objc
}

void foo(int x, int y) {
  id obj = [[NSString alloc] init];
  switch (x) {
    case 0:
      [obj release];
      break;
    case 1:
      //       [obj autorelease];
      break;
    default:
      break;
  }
}
```

Method returns an Objective-C object with a +1 retain count (owning reference)

Object allocated on line 13 is no longer referenced after this point and has a retain count of +1 (object leaked)

https://clang-analyzer.llvm.org

What makes a good static analysis tool?

- Static analysis should be **fast**
  - Don’t hold up development velocity
  - This becomes more important as code scales
- Static analysis should report **few false positives**
  - Otherwise developers will start to ignore warnings and alerts, and quality will decline
- Static analysis should be **continuous**
  - Should be part of your continuous integration pipeline
  - Diff-based analysis is even better -- don’t analyse the entire codebase; just the changes
- Static analysis should be **informative**
  - Messages that help the developer to quickly locate and address the issue
  - Ideally, it should suggest or automatically apply fixes

https://cacm.acm.org/magazines/2010/2/69354-a-few-billion-lines-of-code-later/fulltext

Linters

Cheap, fast, and lightweight static source analysis

https://www.perforce.com/blog/qac/what-lint-code-and-why-linting-important

Linters for Maintainability

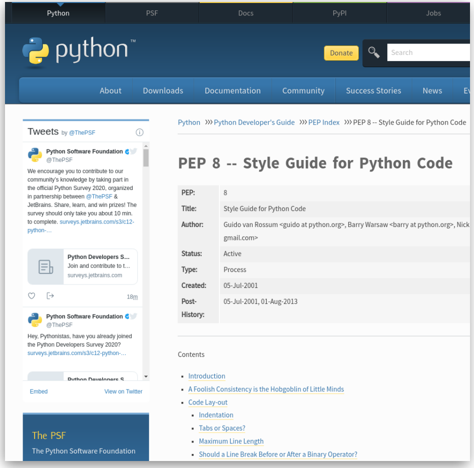
Use linters to improve maintainability

Why? We spend more time reading code than writing it.

- Developers spend most of their time maintaining code
    - Various estimates of the exact %, some as high as 80%
- Code is ownership is usually shared
- The original owner of some code may move on
- Code conventions make it easier for other developers to quickly understand your code

Use Style Guidelines to facilitate communication

- Indentation
- Comments
- Line length
- Naming
- Directory structure
- …



Tweets by @ThePSF

Python Software Foundation
@ThePSF

We encourage you to contribute to our community's knowledge by taking part in the official Python Survey 2020, organized in partnership between @ThePSF & JetBrains. Share, learn, and win prizes! The survey should only take you about 10 min. to complete. surveys.jetbrains.com/s3/c12-python-…

Python Developers S…
Join and contribute to t…
surveys.jetbrains.com

18m

Python Software Foundation
@ThePSF

Hey, Pythonistas, have you already joined the Python Developers Survey 2020? surveys.jetbrains.com/s3/c12-python-…

Python Developers S

Embed View on Twitter

The PSF
The Python Software Foundation

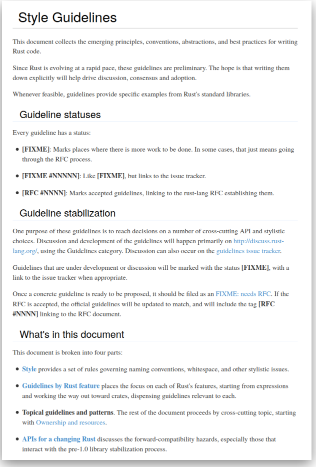
Python ››› Python Developer's Guide ››› PEP Index ››› PEP 8 -- Style Guide for Python Code

PEP 8 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

Contents

- Introduction
- A Foolish Consistency is the Hobgoblin of Little Minds
- Code Lay-out
  - Indentation
  - Tabs or Spaces?
  - Maximum Line Length
  - Should a Line Break Before or After a Binary Operator?



Style Guidelines

This document collects the emerging principles, conventions, abstractions, and best practices for writing Rust code.

Since Rust is evolving at a rapid pace, these guidelines are preliminary. The hope is that writing them down explicitly will help drive discussion, consensus and adoption.

Whenever feasible, guidelines provide specific examples from Rust's standard libraries.

Guideline statuses

Every guideline has a status:

- **[FIXME]**: Marks places where there is more work to be done. In some cases, that just means going through the RFC process.
- **[FIXME #NNNNN]**: Like **[FIXME]**, but links to the issue tracker.
- **[RFC #NNNN]**: Marks accepted guidelines, linking to the rust-lang RFC establishing them.

Guideline stabilization

One purpose of these guidelines is to reach decisions on a number of cross-cutting API and stylistic choices. Discussion and development of the guidelines will happen primarily on http://discuss.rust-lang.org/, using the Guidelines category. Discussion can also occur on the guidelines issue tracker.

Guidelines that are under development or discussion will be marked with the status **[FIXME]**, with a link to the issue tracker when appropriate.

Once a concrete guideline is ready to be proposed, it should be filed as an FIXME: needs RFC. If the RFC is accepted, the official guidelines will be updated to match, and will include the tag **[RFC #NNNN]** linking to the RFC document.

What's in this document

This document is broken into four parts:

- **Style** provides a set of rules governing naming conventions, whitespace, and other stylistic issues.
- **Guidelines by Rust feature** places the focus on each of Rust's features, starting from expressions and working the way out toward crates, dispensing guidelines relevant to each.
- **Topical guidelines and patterns**. The rest of the document proceeds by cross-cutting topic, starting with Ownership and resources.
- **APIs for a changing Rust** discusses the forward-compatibility hazards, especially those that interact with the pre-1.0 library stabilization process.

The Chicago Manual of Style
SEVENTEENTH EDITION
THE ESSENTIAL GUIDE
for Writers, Editors, and Publishers

Guidelines are inherently opinionated, but **consistency** is the important point.
Agree to a set of conventions and stick to them.

https://www.chicagomanualofstyle.org/ | https://google.github.io/styleguide/ | https://www.python.org/dev/peps/pep-0008

Use linters to enforce style guidelines

Don’t rely on manual inspection during code review!

CSi

Java™

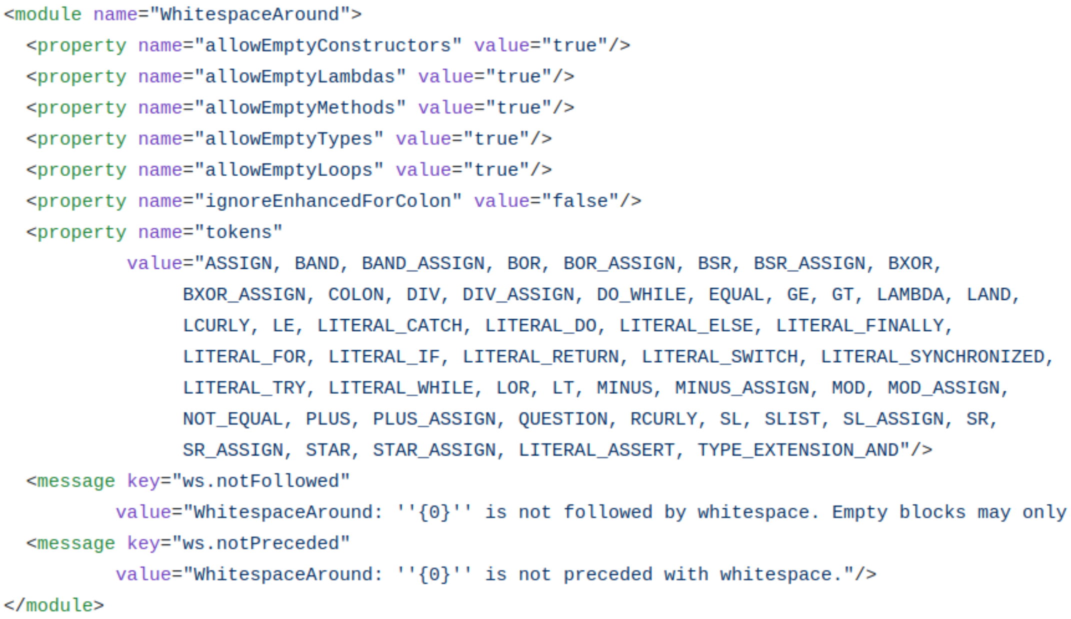
RuboCop

Black

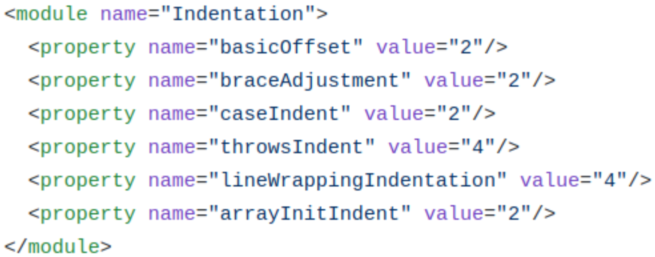
python™

https://checkstyle.sourceforge.io/

Example: CheckStyle



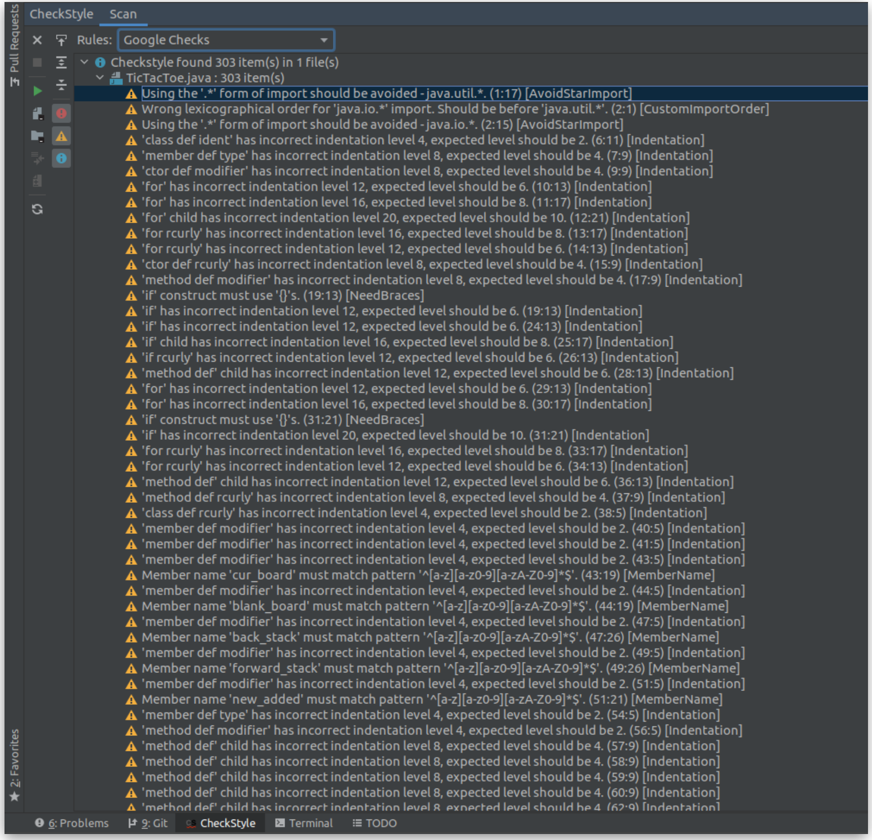
```
<module name="WhitespaceAround">
  <property name="allowEmptyConstructors" value="true"/>
  <property name="allowEmptyLambdas" value="true"/>
  <property name="allowEmptyMethods" value="true"/>
  <property name="allowEmptyTypes" value="true"/>
  <property name="allowEmptyLoops" value="true"/>
  <property name="ignoreEnhancedForColon" value="false"/>
  <property name="tokens"
           value="ASSIGN, BAND, BAND_ASSIGN, BOR, BOR_ASSIGN, BSR, BSR_ASSIGN, BXOR,
                BXOR_ASSIGN, COLON, DIV, DIV_ASSIGN, DO_WHILE, EQUAL, GE, GT, LAMBDA, LAND,
                LCURLY, LE, LITERAL_CATCH, LITERAL_DO, LITERAL_ELSE, LITERAL_FINALLY,
                LITERAL_FOR, LITERAL_IF, LITERAL_RETURN, LITERAL_SWITCH, LITERAL_SYNCHRONIZED,
                LITERAL_TRY, LITERAL_WHILE, LOR, LT, MINUS, MINUS_ASSIGN, MOD, MOD_ASSIGN,
                NOT_EQUAL, PLUS, PLUS_ASSIGN, QUESTION, RCURLY, SL, SLIST, SL_ASSIGN, SR,
                SR_ASSIGN, STAR, STAR_ASSIGN, LITERAL_ASSERT, TYPE_EXTENSION_AND"/>
  <message key="ws.notFollowed"
          value="WhitespaceAround: ''{0}'' is not followed by whitespace. Empty blocks may only
  <message key="ws.notPreceded"
          value="WhitespaceAround: ''{0}'' is not preceded with whitespace."/>
</module>
```



```
<module name="Indentation">
  <property name="basicOffset" value="2"/>
  <property name="braceAdjustment" value="2"/>
  <property name="caseIndent" value="2"/>
  <property name="throwsIndent" value="4"/>
  <property name="lineWrappingIndentation" value="4"/>
  <property name="arrayInitIndent" value="2"/>
</module>
```

...

https://github.com/checkstyle/checkstyle/blob/master/src/main/resources/google_checks.xml



Rules: Google Checks

```
Checkstyle found 303 item(s) in 1 file(s)
  TicTacToe.java : 303 item(s)
    Using the '.*' form of import should be avoided - java.util.*. (1:17) [AvoidStarImport]
    Wrong lexicographical order for 'java.io.*' import. Should be before 'java.util.*'. (2:1) [CustomImportOrder]
    Using the '.*' form of import should be avoided - java.io.*. (2:15) [AvoidStarImport]
    'class def ident' has incorrect indentation level 4, expected level should be 2. (6:11) [Indentation]
    'member def type' has incorrect indentation level 8, expected level should be 4. (7:9) [Indentation]
    'ctor def modifier' has incorrect indentation level 8, expected level should be 4. (9:9) [Indentation]
    'for' has incorrect indentation level 12, expected level should be 6. (10:13) [Indentation]
    'for' has incorrect indentation level 16, expected level should be 8. (11:17) [Indentation]
    'for' child has incorrect indentation level 20, expected level should be 10. (12:21) [Indentation]
    'for rcurly' has incorrect indentation level 16, expected level should be 8. (13:17) [Indentation]
    'for rcurly' has incorrect indentation level 12, expected level should be 6. (14:13) [Indentation]
    'ctor def rcurly' has incorrect indentation level 8, expected level should be 4. (15:9) [Indentation]
    'method def modifier' has incorrect indentation level 8, expected level should be 4. (17:9) [Indentation]
    'if' construct must use '{}'s. (19:13) [NeedBraces]
    'if' has incorrect indentation level 12, expected level should be 6. (19:13) [Indentation]
    'if' has incorrect indentation level 12, expected level should be 6. (24:13) [Indentation]
    'if' child has incorrect indentation level 16, expected level should be 8. (25:17) [Indentation]
    'if rcurly' has incorrect indentation level 12, expected level should be 6. (26:13) [Indentation]
    'method def' child has incorrect indentation level 12, expected level should be 6. (28:13) [Indentation]
    'for' has incorrect indentation level 12, expected level should be 6. (29:13) [Indentation]
    'for' has incorrect indentation level 16, expected level should be 8. (30:17) [Indentation]
    'if' construct must use '{}'s. (31:21) [NeedBraces]
    'if' has incorrect indentation level 20, expected level should be 10. (31:21) [Indentation]
    'for rcurly' has incorrect indentation level 16, expected level should be 8. (33:17) [Indentation]
    'for rcurly' has incorrect indentation level 12, expected level should be 6. (34:13) [Indentation]
    'method def' child has incorrect indentation level 12, expected level should be 6. (36:13) [Indentation]
    'method def rcurly' has incorrect indentation level 8, expected level should be 4. (37:9) [Indentation]
    'class def rcurly' has incorrect indentation level 4, expected level should be 2. (38:5) [Indentation]
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (40:5) [Indentation]
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (41:5) [Indentation]
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (43:5) [Indentation]
    Member name 'cur_board' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*$'. (43:19) [MemberName]
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (44:5) [Indentation]
    Member name 'blank_board' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*$'. (44:19) [MemberName]
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (47:5) [Indentation]
    Member name 'back_stack' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*$'. (47:26) [MemberName]
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (49:5) [Indentation]
    Member name 'forward_stack' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*$'. (49:26) [MemberName]
    'member def modifier' has incorrect indentation level 4, expected level should be 2. (51:5) [Indentation]
    Member name 'new_added' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*$'. (51:21) [MemberName]
    'member def type' has incorrect indentation level 4, expected level should be 2. (54:5) [Indentation]
    'method def modifier' has incorrect indentation level 4, expected level should be 2. (56:5) [Indentation]
    'method def' child has incorrect indentation level 8, expected level should be 4. (57:9) [Indentation]
    'method def' child has incorrect indentation level 8, expected level should be 4. (58:9) [Indentation]
    'method def' child has incorrect indentation level 8, expected level should be 4. (59:9) [Indentation]
    'method def' child has incorrect indentation level 8, expected level should be 4. (60:9) [Indentation]
    'method def' child has incorrect indentation level 8, expected level should be 4. (62:9) [Indentation]
```

```java
@Override
public boolean equals(Object o) {
    if (o == this)
        return true;
```

```java
private Board cur_board;
private Board blank_board;
```

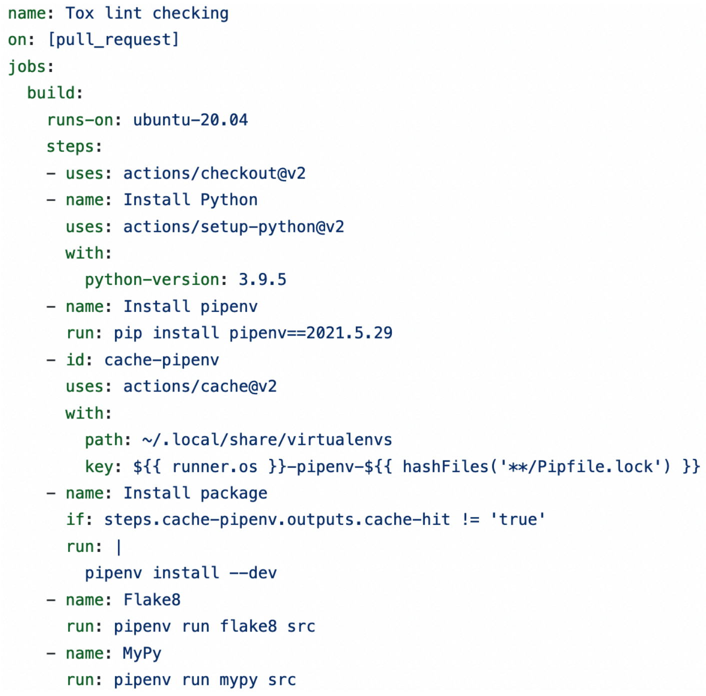
```java
public static void main(String[] args) throws Exception {
    TicTacToe tictactoe = new TicTacToe();
```

Integrate style checking into your CI

```
plugins {
    id 'checkstyle'
}

...

checkstyle {
  ignoreFailures = true
  toolVersion = "6.7"
  sourceSets = [sourceSets.main]
}
```

Travis CI

GitHub Actions

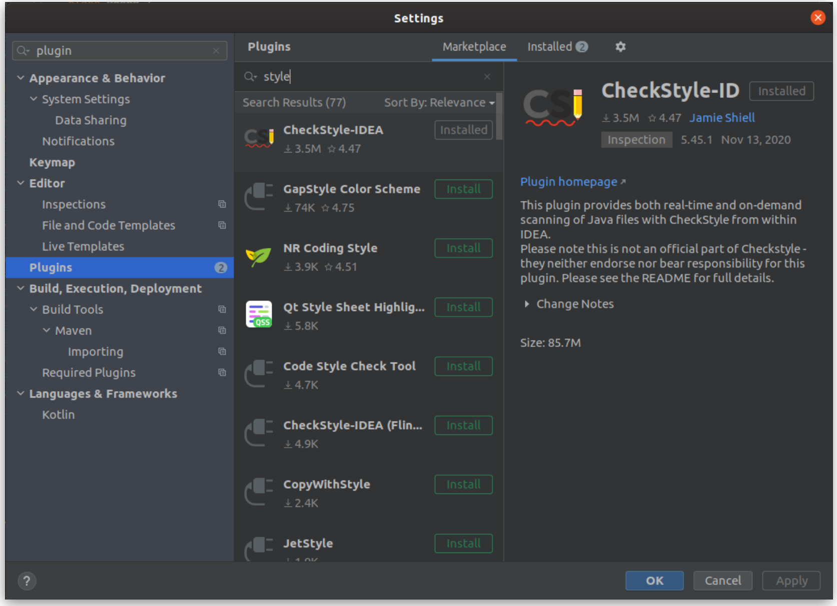


```
name: Tox lint checking
on: [pull_request]
jobs:
  build:
    runs-on: ubuntu-20.04
    steps:
    - uses: actions/checkout@v2
    - name: Install Python
      uses: actions/setup-python@v2
      with:
        python-version: 3.9.5
    - name: Install pipenv
      run: pip install pipenv==2021.5.29
    - id: cache-pipenv
      uses: actions/cache@v2
      with:
        path: ~/.local/share/virtualenvs
        key: ${{ runner.os }}-pipenv-${{ hashFiles('**/Pipfile.lock') }}
    - name: Install package
      if: steps.cache-pipenv.outputs.cache-hit != 'true'
      run: |
        pipenv install --dev
    - name: Flake8
      run: pipenv run flake8 src
    - name: MyPy
      run: pipenv run mypy src
```

https://docs.gradle.org/current/userguide/checkstyle_plugin.html

Automatically reformat your existing code

Developer time is valuable!



style

Search Results (77)

CheckStyle-IDEA	⤓3.5M ☆4.47	Installed
GapStyle Color Scheme	⤓74K ☆4.75	Install
NR Coding Style	⤓3.9K ☆4.51	Install
Qt Style Sheet Highlig...	⤓5.8K	Install
Code Style Check Tool	⤓4.7K	Install
CheckStyle-IDEA (Flin...	⤓4.9K	Install
CopyWithStyle	⤓2.4K	Install
JetStyle		Install

CheckStyle-ID Installed

⤓3.5M ☆4.47 Jamie Shiell

Inspection 5.45.1 Nov 13, 2020

Plugin homepage ↗

This plugin provides both real-time and on-demand scanning of Java files with CheckStyle from within IDEA.
Please note this is not an official part of Checkstyle - they neither endorse nor bear responsibility for this plugin. Please see the README for full details.

Size: 85.7M

Black

https://www.jetbrains.com/help/idea/reformat-and-rearrange-code.html

Take Home Message:
Style is an easy way to improve readability

- Everyone has their own opinion (e.g., tabs vs. spaces)
- Agree to a convention and stick to it
  - Use continuous integration to enforce it
- Use automated tools to fix issues in existing code

S3D

Pattern-Based Static Analyzers

Cheap and fast tools that scan Abstract Syntax Trees for common developer mistakes known as patterns

UNIVERSITY OF MARYLAND 18 56

FindBugs™

because it's easy

RuboCop

clang-tidy

SpotBugs

Bandit

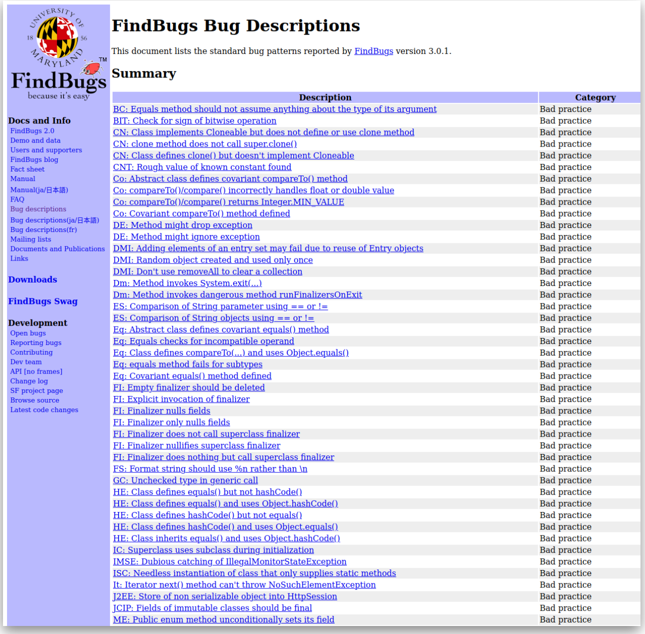
pmd

DON'T SHOOT THE MESSENGER

https://github.com/analysis-tools-dev/static-analysis

SpotBugs

- Bad Practice
- Correctness
- Performance
- Internationalization
- Malicious Code
- Multithreaded Correctness
- Security
- Dodgy Code



Docs and Info
FindBugs 2.0
Demo and data
Users and supporters
FindBugs blog
Fact sheet
Manual
Manual(ja/日本語)
FAQ
Bug descriptions
Bug descriptions(ja/日本語)
Bug descriptions(fr)
Mailing lists
Documents and Publications
Links

Downloads

FindBugs Swag

Development
Open bugs
Reporting bugs
Contributing
Dev team
API [no frames]
Change log
SF project page
Browse source
Latest code changes

FindBugs Bug Descriptions

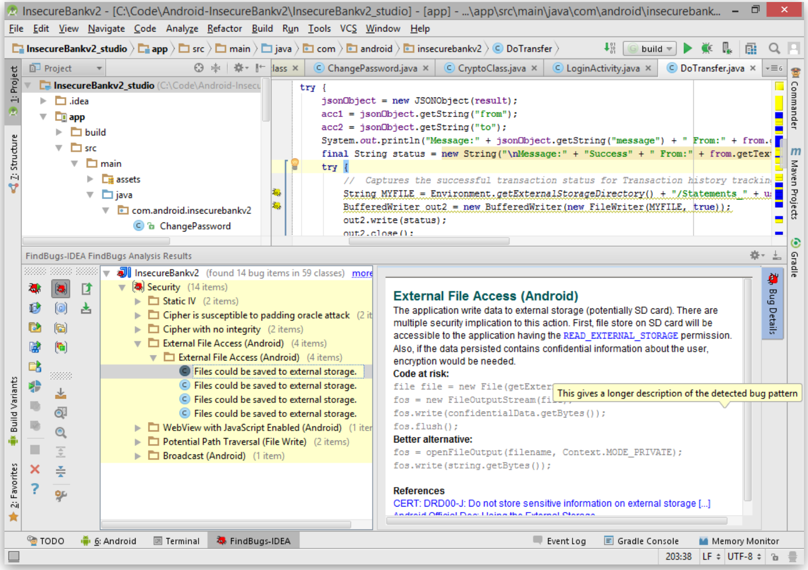
This document lists the standard bug patterns reported by FindBugs version 3.0.1.

Summary

Description	Category
BC: Equals method should not assume anything about the type of its argument	Bad practice
BIT: Check for sign of bitwise operation	Bad practice
CN: Class implements Cloneable but does not define or use clone method	Bad practice
CN: clone method does not call super.clone()	Bad practice
CN: Class defines clone() but doesn't implement Cloneable	Bad practice
CNT: Rough value of known constant found	Bad practice
Co: Abstract class defines covariant compareTo() method	Bad practice
Co: compareTo()/compare() incorrectly handles float or double value	Bad practice
Co: compareTo()/compare() returns Integer.MIN_VALUE	Bad practice
Co: Covariant compareTo() method defined	Bad practice
DE: Method might drop exception	Bad practice
DE: Method might ignore exception	Bad practice
DMI: Adding elements of an entry set may fail due to reuse of Entry objects	Bad practice
DMI: Random object created and used only once	Bad practice
DMI: Don't use removeAll to clear a collection	Bad practice
Dm: Method invokes System.exit(...)	Bad practice
Dm: Method invokes dangerous method runFinalizersOnExit	Bad practice
ES: Comparison of String parameter using == or !=	Bad practice
ES: Comparison of String objects using == or !=	Bad practice
Eq: Abstract class defines covariant equals() method	Bad practice
Eq: Equals checks for incompatible operand	Bad practice
Eq: Class defines compareTo(...) and uses Object.equals()	Bad practice
Eq: equals method fails for subtypes	Bad practice
Eq: Covariant equals() method defined	Bad practice
FI: Empty finalizer should be deleted	Bad practice
FI: Explicit invocation of finalizer	Bad practice
FI: Finalizer nulls fields	Bad practice
FI: Finalizer only nulls fields	Bad practice
FI: Finalizer does not call superclass finalizer	Bad practice
FI: Finalizer nullifies superclass finalizer	Bad practice
FI: Finalizer does nothing but call superclass finalizer	Bad practice
FS: Format string should use %n rather than \n	Bad practice
GC: Unchecked type in generic call	Bad practice
HE: Class defines equals() but not hashCode()	Bad practice
HE: Class defines equals() and uses Object.hashCode()	Bad practice
HE: Class defines hashCode() but not equals()	Bad practice
HE: Class defines hashCode() and uses Object.equals()	Bad practice
HE: Class inherits equals() and uses Object.hashCode()	Bad practice
IC: Superclass uses subclass during initialization	Bad practice
IMSE: Dubious catching of IllegalMonitorStateException	Bad practice
ISC: Needless instantiation of class that only supplies static methods	Bad practice
It: Iterator next() method can't throw NoSuchElementException	Bad practice
J2EE: Store of non serializable object into HttpSession	Bad practice
JCIP: Fields of immutable classes should be final	Bad practice
ME: Public enum method unconditionally sets its field	Bad practice

http://findbugs.sourceforge.net/bugDescriptions.html
https://spotbugs.readthedocs.io/en/latest/bugDescriptions.html

SpotBugs can be extended with plugins



InsecureBankv2 - [C:\Code\Android-InsecureBankv2\InsecureBankv2_studio] - [app] - ...\app\src\main\java\com\android\insecurebank...

```
try {
    jsonObject = new JSONObject(result);
    acc1 = jsonObject.getString("from");
    acc2 = jsonObject.getString("to");
    System.out.println("Message:" + jsonObject.getString("message") + " From:" + from.
    final String status = new String("\nMessage:" + "Success" + " From:" + from.getTex
    try {
        //  Captures the successful transaction status for Transaction history trackin
        String MYFILE = Environment.getExternalStorageDirectory() + "/Statements_" + u
        BufferedWriter out2 = new BufferedWriter(new FileWriter(MYFILE, true));
        out2.write(status);
        out2.close();
```

FindBugs-IDEA FindBugs Analysis Results

- InsecureBankv2 (found 14 bug items in 59 classes) more
  - Security (14 items)
    - Static IV (2 items)
    - Cipher is susceptible to padding oracle attack (2 it
    - Cipher with no integrity (2 items)
    - External File Access (Android) (4 items)
      - External File Access (Android) (4 items)
        - Files could be saved to external storage.
        - Files could be saved to external storage.
        - Files could be saved to external storage.
        - Files could be saved to external storage.
    - WebView with JavaScript Enabled (Android) (1 item
    - Potential Path Traversal (File Write) (2 items)
    - Broadcast (Android) (1 item)

External File Access (Android)

The application write data to external storage (potentially SD card). There are multiple security implication to this action. First, file store on SD card will be accessible to the application having the READ_EXTERNAL_STORAGE permission. Also, if the data persisted contains confidential information about the user, encryption would be needed.

Code at risk:

```
file file = new File(getExter
fos = new FileOutputStream(fi
fos.write(confidentialData.getBytes());
fos.flush();
```

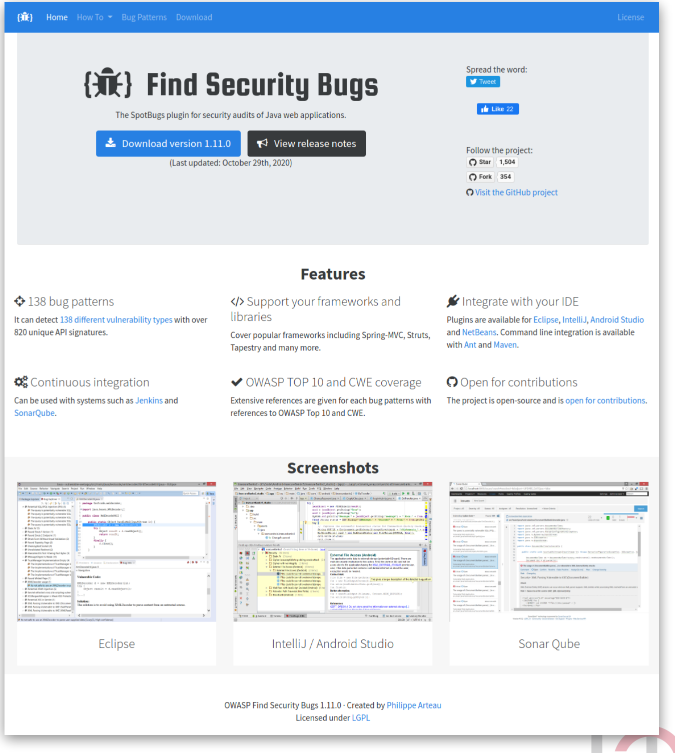
Better alternative:

```
fos = openFileOutput(filename, Context.MODE_PRIVATE);
fos.write(string.getBytes());
```

References

CERT: DRD00-J: Do not store sensitive information on external storage [...]

This gives a longer description of the detected bug pattern



Find Security Bugs

The SpotBugs plugin for security audits of Java web applications.

Download version 1.11.0

View release notes

(Last updated: October 29th, 2020)

Spread the word:

Follow the project:

Visit the GitHub project

Features

138 bug patterns

It can detect 138 different vulnerability types with over 820 unique API signatures.

Support your frameworks and libraries

Cover popular frameworks including Spring-MVC, Struts, Tapestry and many more.

Integrate with your IDE

Plugins are available for Eclipse, IntelliJ, Android Studio and NetBeans. Command line integration is available with Ant and Maven.

Continuous integration

Can be used with systems such as Jenkins and SonarQube.

OWASP TOP 10 and CWE coverage

Extensive references are given for each bug patterns with references to OWASP Top 10 and CWE.

Open for contributions

The project is open-source and is open for contributions.

Screenshots

Eclipse

IntelliJ / Android Studio

Sonar Qube

OWASP Find Security Bugs 1.11.0 · Created by Philippe Arteau

Licensed under LGPL

https://find-sec-bugs.github.io/

Bad Practice:

```java
String x = new String("Foo");
String y = new String("Foo");

if (x == y) {
  System.out.println("x and y are the same!");
} else {
  System.out.println("x and y are different!");
}
```

Bad Practice: `ES_COMPARING_STRINGS_WITH_EQ`

Comparing strings with ==

```
String x = new String("Foo");
String y = new String("Foo");

if (x == y) {
if (x.equals(y)) {
  System.out.println("x and y are the same!");
} else {
  System.out.println("x and y are different!");
}
```

Performance:

```
public static String repeat(String string, int times)
{
  String output = string;
  for (int i = 1; i < times; ++i) {
    output = output + string;
  }
  return output;
}
```

Performance: `SBSC_USE_STRINGBUFFER_CONCATENATION`

Method concatenates strings using + in a loop

```java
public static String repeat(String string, int times)
{
  String output = string;
  for (int i = 1; i < times; ++i) {
    output = output + string;
  }
  return output;
}
```

The method seems to be building a String using concatenation in a loop. In each iteration, the String is converted to a StringBuffer/StringBuilder, appended to, and converted back to a String. **This can lead to a cost quadratic in the number of iterations, as the growing string is recopied in each iteration.**

Performance: `SBSC_USE_STRINGBUFFER_CONCATENATION`

Method concatenates strings using + in a loop

```java
public static String repeat(String string, int times)
{
  StringBuffer output = new StringBuffer(string);
  for (int i = 1; i < times; ++i) {
    output.append(string);
  }
  return output.toString();
}
```

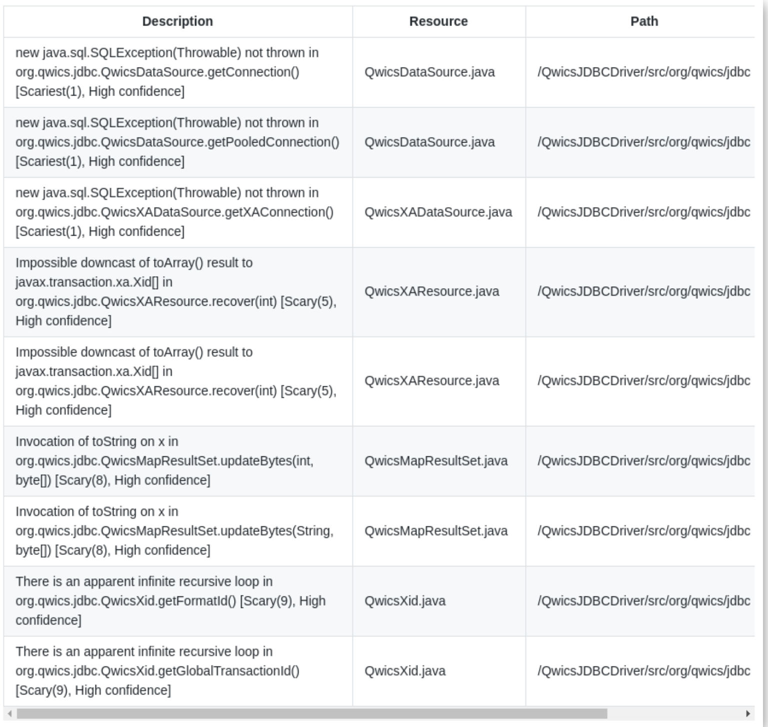
Performance: `SBSC_USE_STRINGBUFFER_CONCATENATION`

Method concatenates strings using + in a loop

```java
public static String repeat(String string, int times)
{
  int length = string.length() * times;
  StringBuffer output = new StringBuffer(length);
  for (int i = 0; i < times; ++i) {
    output.append(string);
  }
  return output.toString();
}
```

Correctness: Lots of issues here!

```
public class QwicsXid implements Xid {
  private byte[] globalTransactionId;
  private byte[] branchQualifier;
  private int formatId;
  ...

  @Override
  public byte[] getBranchQualifier() {
    return this.branchQualifier;
  }
  @Override
  public int getFormatId() {
    return this.getFormatId();
  }
  @Override
  public byte[] getGlobalTransactionId() {
    return this.getGlobalTransactionId();
  }
}
```



Description	Resource	Path
new java.sql.SQLException(Throwable) not thrown in org.qwics.jdbc.QwicsDataSource.getConnection() [Scariest(1), High confidence]	QwicsDataSource.java	/QwicsJDBCDriver/src/org/qwics/jdbc
new java.sql.SQLException(Throwable) not thrown in org.qwics.jdbc.QwicsDataSource.getPooledConnection() [Scariest(1), High confidence]	QwicsDataSource.java	/QwicsJDBCDriver/src/org/qwics/jdbc
new java.sql.SQLException(Throwable) not thrown in org.qwics.jdbc.QwicsXADataSource.getXAConnection() [Scariest(1), High confidence]	QwicsXADataSource.java	/QwicsJDBCDriver/src/org/qwics/jdbc
Impossible downcast of toArray() result to javax.transaction.xa.Xid[] in org.qwics.jdbc.QwicsXAResource.recover(int) [Scary(5), High confidence]	QwicsXAResource.java	/QwicsJDBCDriver/src/org/qwics/jdbc
Impossible downcast of toArray() result to javax.transaction.xa.Xid[] in org.qwics.jdbc.QwicsXAResource.recover(int) [Scary(5), High confidence]	QwicsXAResource.java	/QwicsJDBCDriver/src/org/qwics/jdbc
Invocation of toString on x in org.qwics.jdbc.QwicsMapResultSet.updateBytes(int, byte[]) [Scary(8), High confidence]	QwicsMapResultSet.java	/QwicsJDBCDriver/src/org/qwics/jdbc
Invocation of toString on x in org.qwics.jdbc.QwicsMapResultSet.updateBytes(String, byte[]) [Scary(8), High confidence]	QwicsMapResultSet.java	/QwicsJDBCDriver/src/org/qwics/jdbc
There is an apparent infinite recursive loop in org.qwics.jdbc.QwicsXid.getFormatId() [Scary(9), High confidence]	QwicsXid.java	/QwicsJDBCDriver/src/org/qwics/jdbc
There is an apparent infinite recursive loop in org.qwics.jdbc.QwicsXid.getGlobalTransactionId() [Scary(9), High confidence]	QwicsXid.java	/QwicsJDBCDriver/src/org/qwics/jdbc

https://github.com/pbrune1973/qwics/blob/3b4ec904468eaed89ea890b5ae97ebaaa8e4a3e6/workspace/QwicsJDBCDriver/src/org/qwics/jdbc/QwicsXid.java
https://github.com/pbrune1973/qwics/issues/7

Correctness:

```
@Override
public Connection getConnection() throws SQLException {
  QwicsConnection con = new QwicsConnection(host, port);
  try {
    con.open();
  } catch (Exception e) {
    new SQLException(e);
  }
  return con;
}
```

https://github.com/pbrune1973/qwics/blob/3b4ec904468eaed89ea890b5ae97ebaaa8e4a3e6/workspace/QwicsJDBCDriver/src/org/qwics/jdbc/QwicsXid.java

https://github.com/pbrune1973/qwics/issues/7

Correctness:

```
@Override
public Connection getConnection() throws SQLException {
  QwicsConnection con = new QwicsConnection(host, port);
  try {
    con.open();
  } catch (Exception e) {
    throw new SQLException(e);
  }
  return con;
}
```

https://github.com/pbrune1973/qwics/blob/3b4ec904468eaed89ea890b5ae97ebaaa8e4a3e6/workspace/QwicsJDBCDriver/src/org/qwics/jdbc/QwicsXid.java
https://github.com/pbrune1973/qwics/issues/7

What are some of the problems with SpotBugs?

Google: Move static checks to the compiler

Developers can ignore warnings, but they can't ignore build errors

clang-tidy

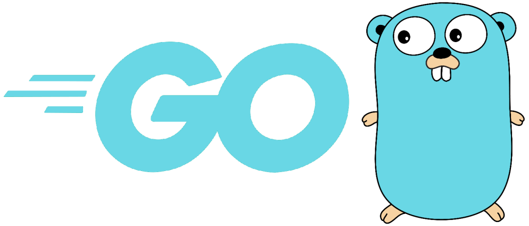
C++ C

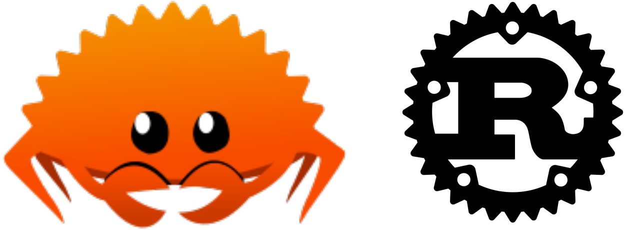
Error Prone

https://cacm.acm.org/magazines/2018/4/226371-lessons-from-building-static-analysis-tools-at-google/fulltext

New languages have embraced the same idea

Code smells will cause the build to fail (e.g., dead code)





Challenges

- The analysis must produce zero false positives
  - Otherwise developers won't be able to build the code!
- The analysis needs to be really fast
  - Ideally < 100 ms
  - If it takes longer, developers will become irritated and lose productivity
- You can't just "turn on" a particular check
  - Every instance where that check fails will prevent existing code from building
  - There could be thousands of violations for a single check across large codebases

Challenges

- The analysis must produce zero false positives
  - Otherwise developers won’t be able to build the code!
- The analysis needs to be really fast
  - Ideally < 100 ms
  - If it takes longer, developers will become irritated and lose productivity
- You can’t just “turn on” a particular check
  - Every instance where that check fails will prevent existing code from building
  - There could be thousands of violations for a single check across large codebases

Solution: Automatically patch existing bugs

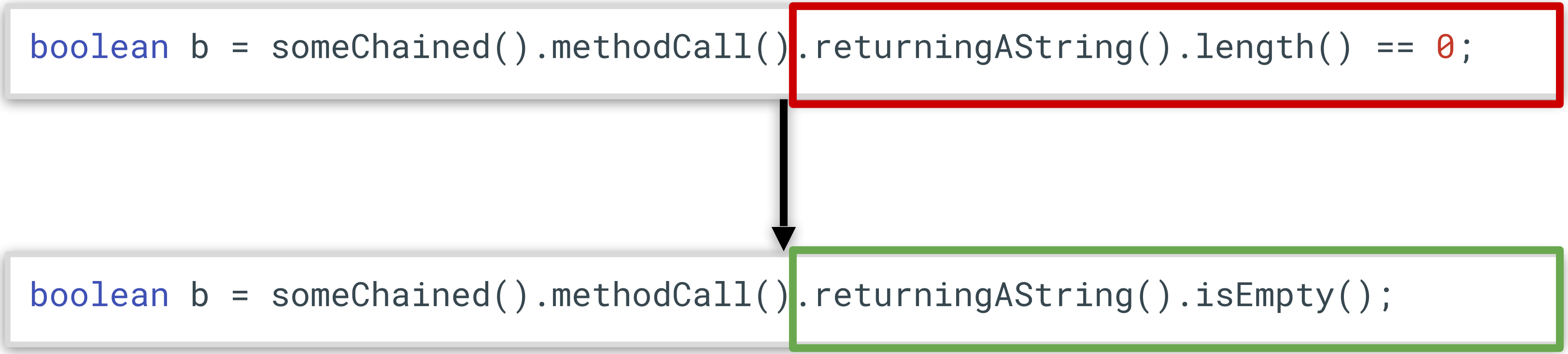
```java
public class StringIsEmpty {
  @BeforeTemplate
  boolean equalsEmptyString(String string) {
    return string.equals("");
  }

  @BeforeTemplate
  boolean lengthEquals0(String string) {
    return string.length() == 0;
  }

  @AfterTemplate
  @AlsoNegation
  boolean optimizedMethod(String string) {
    return string.isEmpty();
  }
}
```

@BeforeTemplate finds String expressions that match the body of the method.

@AfterTemplate rewrites matching String expressions to match the body of the method.

https://errorprone.info/docs/refaster

Solution: Automatically patch existing bugs



```java
boolean b = someChained().methodCall().returningAString().length() == 0;
```

```java
boolean b = someChained().methodCall().returningAString().isEmpty();
```

https://errorprone.info/docs/refaster

Summary: Linters

- Linters are cheap and fast static analysis tools!
- Style checkers can improve readability of code
- Pattern-based bug detectors catch common developer mistakes
    - Code smells, performance issues, correctness, …
    - They don't know the intent of the program, leading to occasional false positives
    - They reveal issues that are genuine, but which we don't sufficiently care about
    - The best tools automatically fix detected issues
    - Each developer mistake needs its own analyzer / AST checker
    - They *complement* but don't *replace* testing

Java Checker Framework

Uses annotations to detect common errors

- Uses a conservative analysis to prove the absence of certain defects *
  - Null pointer errors, uninitialized fields, certain liveness issues, information leaks, SQL injections, bad regular expressions, incorrect physical units, bad format strings, …
  - C.f. SpotBugs which makes no safety guarantees
  - Assuming that code is annotated and those annotations are correct
- Uses annotations to enhance Java’s type system

CHECKER framework

https://checkerframework.org/

Annotations can be applied to types and declarations

```
// return value
@InternedString intern() { ... }

// parameter
int compareTo(@NonNullString other) { ... }

// receiver ("this" parameter)
String toString(@TaintedMyClass this) { ... }

// generics: non-null list of interned Strings
@NonNullList<@InternedString> messages;

// arrays:  non-null array of interned Strings
@InternedString @NonNull[] messages;

// cast
myDate = (@InitializedDate) beingConstructed;
```

Detecting null pointer exceptions

- **@Nullable** indicates that an expression may be null
- **@NonNull** indicates that an expression must never be null
  - Rarely used because @NonNull is assumed by default
  - See documentation for other nullness annotations
- Guarantees that expressions annotated with @NonNull will **never** evaluate to null, forbids other expressions from being dereferenced

https://checkerframework.org/manual/#nullness-annotations

```
import org.checkerframework.checker.nullness.qual.*;

public class NullnessExampleWithWarnings {
    public void example() {
        @NonNull String foo = "foo";
        String bar = null;

        foo = bar;
    }
}
```

```java
import org.checkerframework.checker.nullness.qual.*;

public class NullnessExampleWithWarnings {
    public void example() {
        @NonNull String foo = "foo";
        String bar = null;

        foo = bar;
    }
}
```

`@Nullable` is applied by default

```java
import org.checkerframework.checker.nullness.qual.*;

public class NullnessExampleWithWarnings {
    public void example() {
        @NonNull String foo = "foo";
        String bar = null;

      🐞 foo = bar;
    }
}
```

`@Nullable` is applied by default

Error: [assignment.type.incompatible] incompatible types in assignment.
  found   : @Initialized **@Nullable** String
  required: @UnknownInitialization **@NonNull** String

```java
import org.checkerframework.checker.nullness.qual.*;

public class NullnessExampleWithWarnings {
  public void example() {
    @NonNull String foo = "foo";
    String bar = null;   // @Nullable

    if (bar != null) {
      foo = bar;
    }
  }
}
```

bar is refined to `@NonNull`

Is there a bug?

```
public String getDay(int dayIndex) {
  String day = null;
  switch (dayIndex) {
    case 0: day = "Monday";
    case 1: day = "Tuesday";
    case 2: day = "Wednesday";
    case 3: day = "Thursday";
  }
  return day;
}

public void example() {
  @NonNull String dayName = getDay(4);
  System.out.println("Today is " + dayName);
}
```

Is there a bug? Yes.

```
public String getDay(int dayIndex) {
  String day = null;
  switch (dayIndex) {
    case 0: day = "Monday";
    case 1: day = "Tuesday";
    case 2: day = "Wednesday";
    case 3: day = "Thursday";
  }
  return day;
}

public void example() {
  @NonNull String dayName = getDay(4);
  System.out.println("Today is " + dayName);
}
```

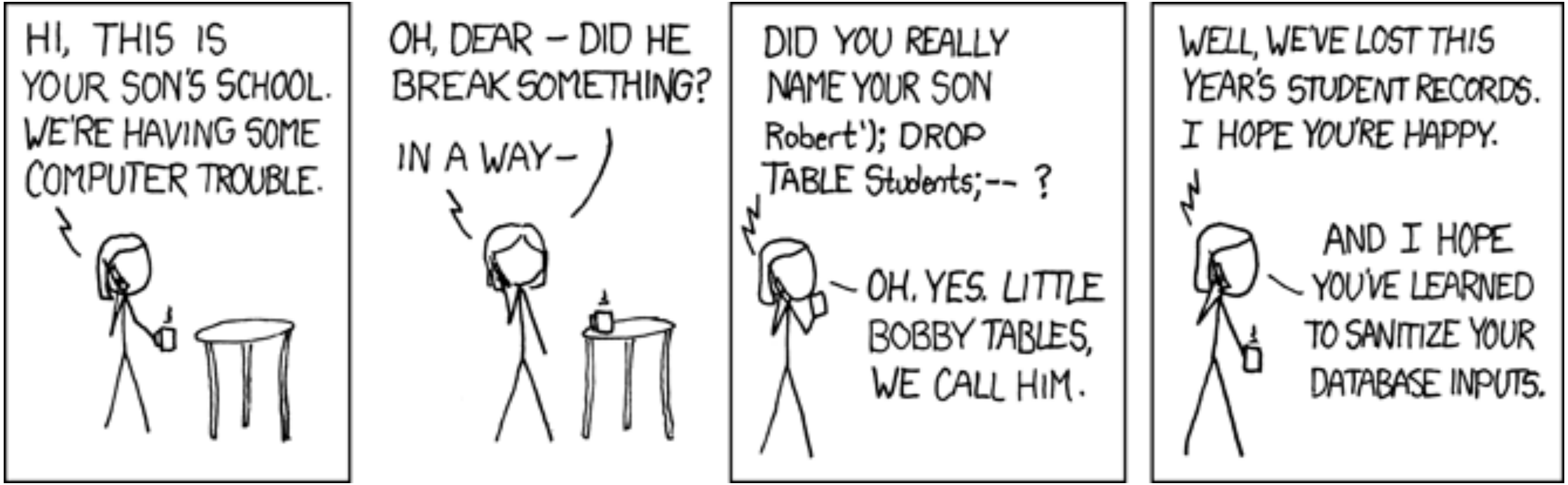
```
Error: [return.type.incompatible] incompatible types in return.
   type of expression: @Initialized @Nullable String
   method return type: @Initialized @NonNull String
```

Taint Analysis

Prevents untrusted (tainted) data from reaching sensitive locations (sinks)

- Tracks flow of sensitive information through the program
- Tainted inputs come from arbitrary, possibly malicious sources
    - User inputs, unvalidated data
- Using tainted inputs may have dangerous consequences
    - Program crash, data corruption, leak private data, etc.
- We need to check that inputs are sanitized before reaching sensitive locations

Classic Example: SQL Injection


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

https://xkcd.com/327

Classic Example: SQL Injection

```
void processRequest() {
  String input = getUserInput();
  String query = "SELECT ... " + input;
  executeQuery(query);
}
```

Classic Example: SQL Injection

Tainted input arrives from an untrusted source

```
void processRequest() {
  String input = getUserInput();
  String query = "SELECT ... " + input;
  executeQuery(query);
}
```

Tainted input flows to a sensitive sink

Classic Example: SQL Injection

Taint is removed by sanitizing the data

```
void processRequest() {
  String input = getUserInput();
  input = sanitizeInput(input);
  String query = "SELECT ... " + input;
  executeQuery(query);
}
```

We can now safely execute query on untainted data

Taint Checker: `@Tainted` and `@Untainted`

```
void processRequest() {
  @Tainted String input = getUserInput();
  executeQuery(input);
}

public void executeQuery(@Untainted String input) {
  // ...
}

@Untainted public String validate(String userInput) {
  // ...
}
```

Taint Checker: `@Tainted` and `@Untainted`

```
void processRequest() {
  @Tainted String input = getUserInput();
  executeQuery(input);
}

public void executeQuery(@Untainted String input) {
  // ...
}

@Untainted public String validate(String userInput) {
  // ...
}
```

Indicates that data is tainted

Argument ***must*** be untainted

Guarantees that return value is untainted

Taint Checker: `@Tainted` and `@Untainted`

```
void processRequest() {
  @Tainted String input = getUserInput();
  executeQuery(input);
}

public void executeQuery(@Untainted String input) {
  // ...
}

@Untainted public String validate(String userInput) {
  // ...
}
```

Indicates that data is tainted

Argument ***must*** be untainted

Guarantees that return value is untainted

Does this compile?

```
void processRequest() {
  @Tainted String input = getUserInput();
  input = validate(input);
  executeQuery(input);
}

public void executeQuery(@Untainted String input) {
  // ...
}

@Untainted public String validate(String userInput) {
  // ...
}
```

Input becomes `@Untainted`

Does this program compile?

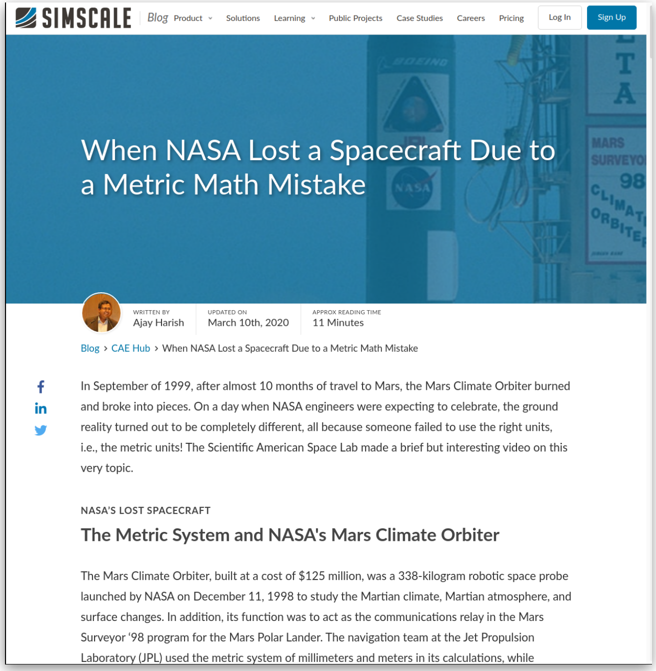
```java
void processRequest() {
  @Tainted String input = getUserInput();
  if (input.equals("little bobby drop tables")) {
    input = validate(input);
  }
  executeQuery(input);
}
```

Does this program compile? No.

```
void processRequest() {
  @Tainted String input = getUserInput();
  if (input.equals("little bobby drop tables")) {
    input = validate(input); // @Untainted
  }
  executeQuery(input); // @Tainted
}
```

AUTH
NASA
LOCKHEED
"METRIC, ENGLISH, WHATEVER..."

Remember the Mars Climate Orbiter incident from 1999?



When NASA Lost a Spacecraft Due to a Metric Math Mistake

WRITTEN BY Ajay Harish

UPDATED ON March 10th, 2020

APPROX READING TIME 11 Minutes

Blog › CAE Hub › When NASA Lost a Spacecraft Due to a Metric Math Mistake

In September of 1999, after almost 10 months of travel to Mars, the Mars Climate Orbiter burned and broke into pieces. On a day when NASA engineers were expecting to celebrate, the ground reality turned out to be completely different, all because someone failed to use the right units, i.e., the metric units! The Scientific American Space Lab made a brief but interesting video on this very topic.

NASA'S LOST SPACECRAFT

The Metric System and NASA's Mars Climate Orbiter

The Mars Climate Orbiter, built at a cost of $125 million, was a 338-kilogram robotic space probe launched by NASA on December 11, 1998 to study the Martian climate, Martian atmosphere, and surface changes. In addition, its function was to act as the communications relay in the Mars Surveyor '98 program for the Mars Polar Lander. The navigation team at the Jet Propulsion Laboratory (JPL) used the metric system of millimeters and meters in its calculations, while

NASA's Mars Climate Orbiter (cost of $327 million) was lost because of a discrepancy between use of metric unit Newtons and imperial measure Pound-force.

https://www.simscale.com/blog/2017/12/nasa-mars-climate-orbiter-metric/

Units Checker identifies physical unit inconsistencies

- Guarantees that operations are performed on the same kinds and units
- Kind annotations
    - `@Acceleration, @Angle, @Area, @Current, @Length, @Luminance, @Mass, @Speed, @Substance, @Temperature, @Time`
- SI unit annotation
    - `@m, @km, @mm, @kg, @mPERs, @mPERs2, @radians, @degrees, @A, ...`

https://www.nist.gov/pml/weights-and-measures/metric-si/si-units

```
import static org.checkerframework.checker.units.UnitsTools.m;
import static org.checkerframework.checker.units.UnitsTools.mPERs;
import static org.checkerframework.checker.units.UnitsTools.s;

void demo() {
  @m int x;
  x = 5 * m;

  @m int meters = 5 * m;
  @s int seconds = 2 * s;

  @mPERs int speed = meters / seconds;
  @m int foo = meters + seconds;
  @s int bar = seconds - meters;
}
```

```
import static org.checkerframework.checker.units.UnitsTools.m;
import static org.checkerframework.checker.units.UnitsTools.mPERs;
import static org.checkerframework.checker.units.UnitsTools.s;

void demo() {
  @m int x;
  x = 5 * m;

  @m int meters = 5 * m;
  @s int seconds = 2 * s;

  @mPERs int speed = meters / seconds;
  @m int foo = meters + seconds;
  @s int bar = seconds - meters;
}
```

`@m` indicates that `x` represents meters

To assign a unit, multiply appropriate unit constant from `UnitTools`

Does this program compile?

```
import static org.checkerframework.checker.units.UnitsTools.m;
import static org.checkerframework.checker.units.UnitsTools.mPERs;
import static org.checkerframework.checker.units.UnitsTools.s;

void demo() {
  @m int x;
  x = 5 * m;

  @m int meters = 5 * m;
  @s int seconds = 2 * s;

  @mPERs int speed = meters / seconds;
  @m int foo = meters + seconds;
  @s int bar = seconds - meters;
}
```

`@m` indicates that `x` represents meters

To assign a unit, multiply appropriate unit constant from `UnitTools`

Does this program compile? No.

```
import static org.checkerframework.checker.units.UnitsTools.m;
import static org.checkerframework.checker.units.UnitsTools.mPERs;
import static org.checkerframework.checker.units.UnitsTools.s;

void demo() {
  @m int x;
  x = 5 * m;

  @m int meters = 5 * m;
  @s int seconds = 2 * s;

  @mPERs int speed = meters / seconds;
  @m int foo = meters + seconds;
  @s int bar = seconds - meters;
}
```

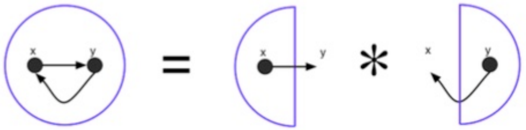
Addition and subtraction between meters and seconds is physically meaningless

Checker Framework: Limitations

- **Can only analyze code that is annotated**
  - Requires that dependent libraries are also annotated
  - Can be tricky, but not impossible, to retrofit annotations into existing codebases
- Only considers the signature and annotations of methods
  - Doesn’t look at the implementation of methods that are being called
- Dynamically generated code
  - Spring Framework
- Can produce false positives!
  - Byproduct of necessary approximations

Infer: What if we didn’t need annotations?

- Focused on memory safety bugs
  - Null pointer dereferences, memory leaks, resource leaks, …
- Compositional interprocedural reasoning
  - Based on separation logic and bi-abduction
- Scalable and fast
  - Can run incremental analysis on changed code
- **Does not require annotations**
- **Supports multiple languages**
  - Java, C, C++, Objective-C
  - Programs are compiled to an intermediate representation

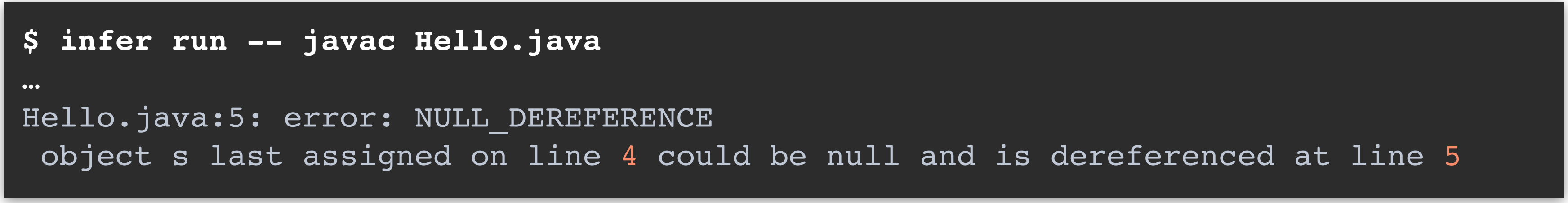


https://fbinfer.com/

https://engineering.fb.com/2017/09/06/android/finding-inter-procedural-bugs-at-scale-with-infer-static-analyzer/

Infer: Hello World!

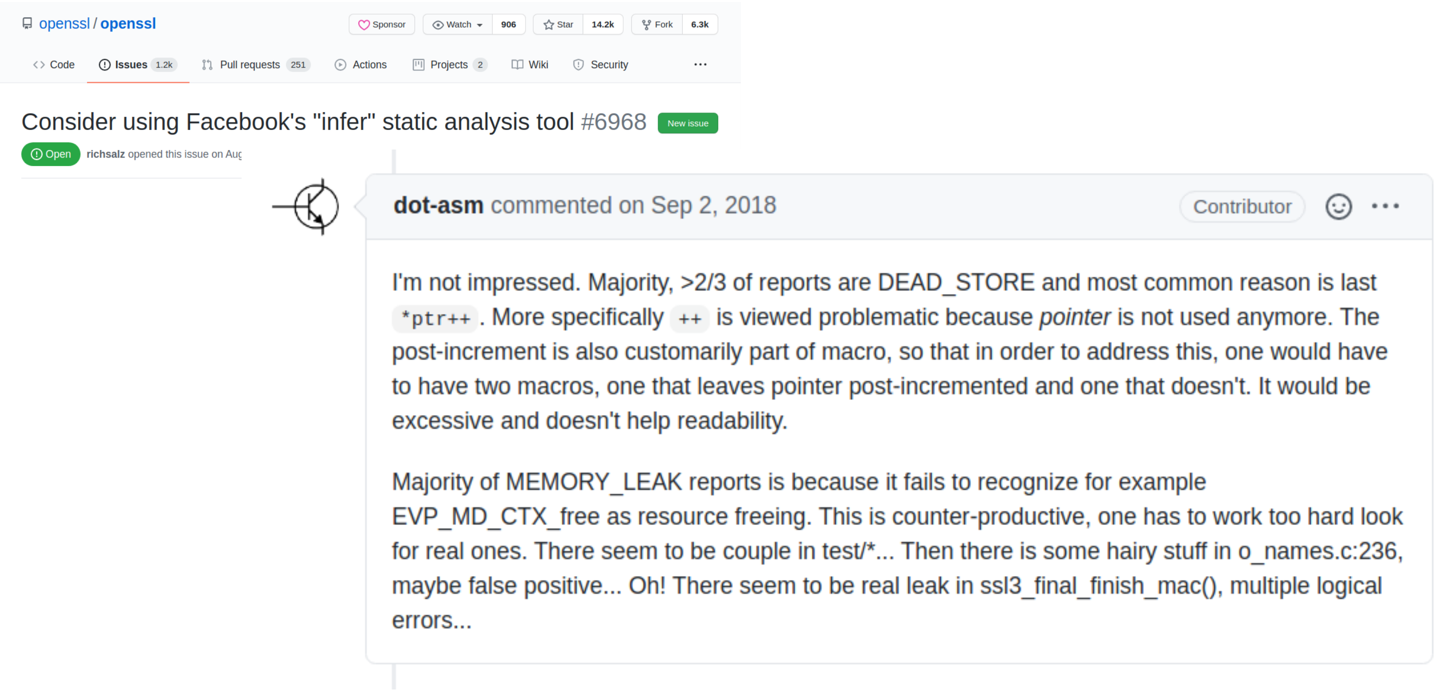
```java
// Hello.java
class Hello {
  int test() {
    String s = null;
    return s.length();
  }
}
```



```
$ infer run -- javac Hello.java
…
Hello.java:5: error: NULL_DEREFERENCE
 object s last assigned on line 4 could be null and is dereferenced at line 5
```

https://fbinfer.com/docs/hello-world

Beware of the inevitable false positives!



Consider using Facebook's "infer" static analysis tool #6968

richsalz opened this issue on Auç

dot-asm commented on Sep 2, 2018

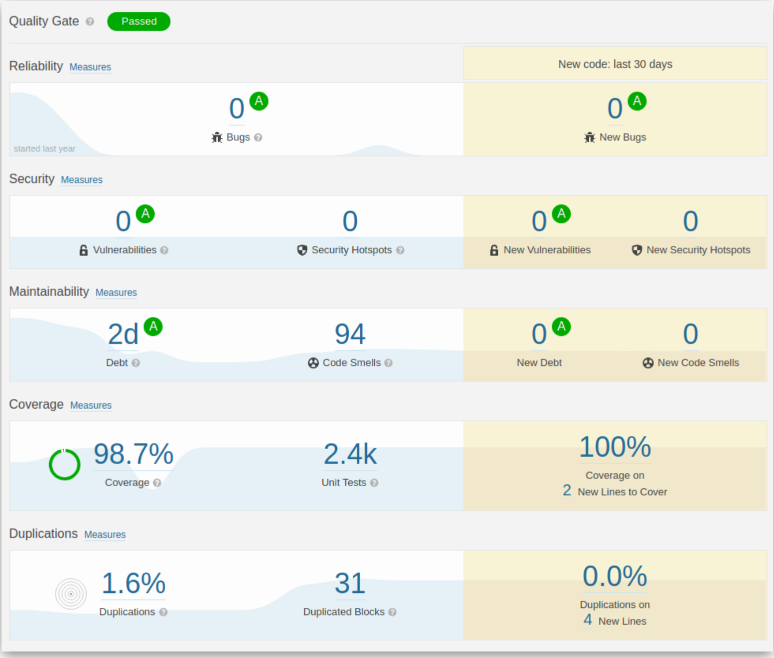
I'm not impressed. Majority, >2/3 of reports are DEAD_STORE and most common reason is last `*ptr++`. More specifically `++` is viewed problematic because *pointer* is not used anymore. The post-increment is also customarily part of macro, so that in order to address this, one would have to have two macros, one that leaves pointer post-incremented and one that doesn't. It would be excessive and doesn't help readability.

Majority of MEMORY_LEAK reports is because it fails to recognize for example EVP_MD_CTX_free as resource freeing. This is counter-productive, one has to work too hard look for real ones. There seem to be couple in test/*... Then there is some hairy stuff in o_names.c:236, maybe false positive... Oh! There seem to be real leak in ssl3_final_finish_mac(), multiple logical errors...

https://github.com/openssl/openssl/issues/6968

Analysis Dashboards

A holistic approach to quality: SonarQube



Quality Gate Passed

Reliability Measures

New code: last 30 days

0 A Bugs

started last year

0 A New Bugs

Security Measures

0 A Vulnerabilities

0 Security Hotspots

0 A New Vulnerabilities

0 New Security Hotspots

Maintainability Measures

2d A Debt

94 Code Smells

0 A New Debt

0 New Code Smells

Coverage Measures

98.7% Coverage

2.4k Unit Tests

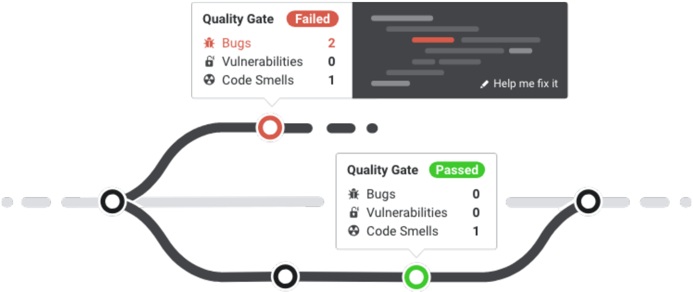
100% Coverage on 2 New Lines to Cover

Duplications Measures

1.6% Duplications

31 Duplicated Blocks

0.0% Duplications on 4 New Lines



Quality Gate Failed

Bugs	2
Vulnerabilities	0
Code Smells	1

Help me fix it

Quality Gate Passed

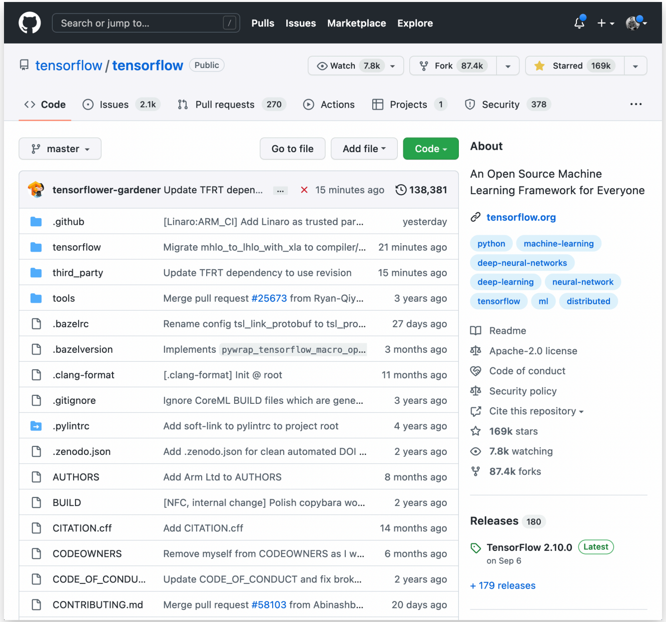
Bugs	0
Vulnerabilities	0
Code Smells	1

sonarcloud

sonarqube

https://sonarcloud.io

Let’s look at a real project using SonarQube: TensorFlow



tensorflower-gardener Update TFRT depen... ... ✕ 15 minutes ago 138,381

.github	[Linaro:ARM_CI] Add Linaro as trusted par...	yesterday
tensorflow	Migrate mhlo_to_lhlo_with_xla to compiler/...	21 minutes ago
third_party	Update TFRT dependency to use revision	15 minutes ago
tools	Merge pull request #25673 from Ryan-Qiy...	3 years ago
.bazelrc	Rename config tsl_link_protobuf to tsl_pro...	27 days ago
.bazelversion	Implements `pywrap_tensorflow_macro_op...`	3 months ago
.clang-format	[.clang-format] Init @ root	11 months ago
.gitignore	Ignore CoreML BUILD files which are gene...	3 years ago
.pylintrc	Add soft-link to pylintrc to project root	4 years ago
.zenodo.json	Add .zenodo.json for clean automated DOI ...	2 years ago
AUTHORS	Add Arm Ltd to AUTHORS	8 months ago
BUILD	[NFC, internal change] Polish copybara wo...	2 years ago
CITATION.cff	Add CITATION.cff	14 months ago
CODEOWNERS	Remove myself from CODEOWNERS as I w...	6 months ago
CODE_OF_CONDU...	Update CODE_OF_CONDUCT and fix brok...	2 years ago
CONTRIBUTING.md	Merge pull request #58103 from Abinashb...	20 days ago

About

An Open Source Machine Learning Framework for Everyone

tensorflow.org

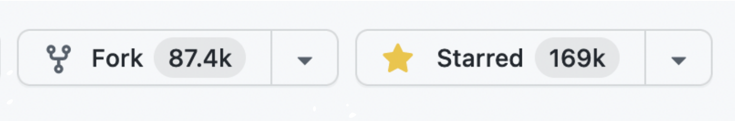
python machine-learning deep-neural-networks deep-learning neural-network tensorflow ml distributed

Readme
Apache-2.0 license
Code of conduct
Security policy
Cite this repository
169k stars
7.8k watching
87.4k forks

Releases 180

TensorFlow 2.10.0 Latest
on Sep 6

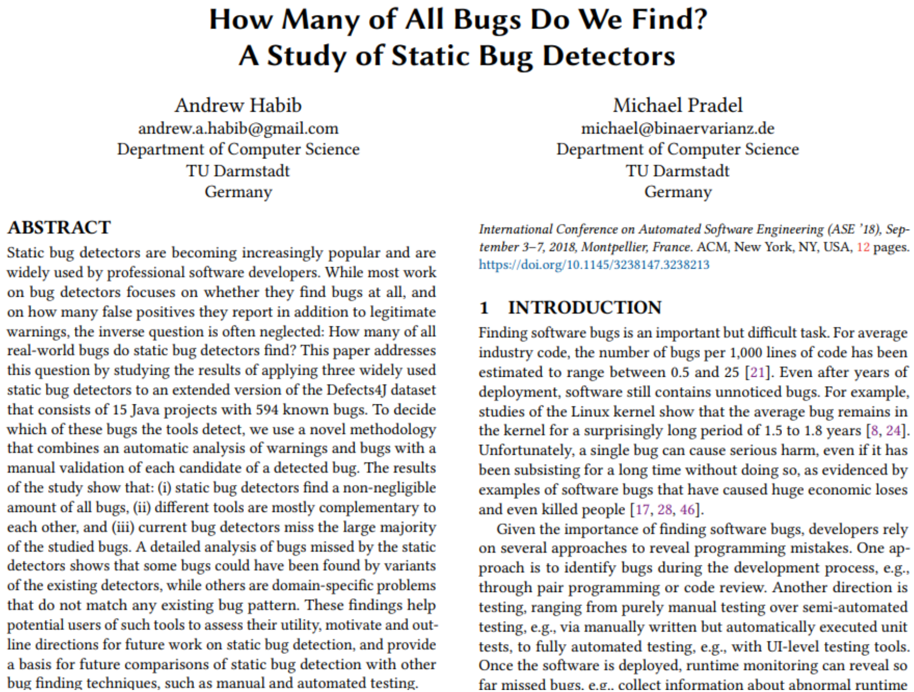
+ 179 releases



https://sonarcloud.io/summary/overall?id=htefera_Tensorflow

What analysis tools should I use?

The best QA strategies employ a combination of tools



How Many of All Bugs Do We Find? A Study of Static Bug Detectors

Andrew Habib
andrew.a.habib@gmail.com
Department of Computer Science
TU Darmstadt
Germany

Michael Pradel
michael@binaervarianz.de
Department of Computer Science
TU Darmstadt
Germany

ABSTRACT

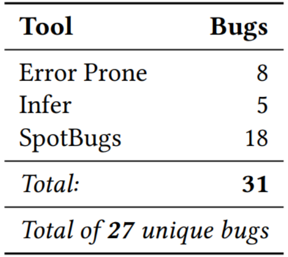
Static bug detectors are becoming increasingly popular and are widely used by professional software developers. While most work on bug detectors focuses on whether they find bugs at all, and on how many false positives they report in addition to legitimate warnings, the inverse question is often neglected: How many of all real-world bugs do static bug detectors find? This paper addresses this question by studying the results of applying three widely used static bug detectors to an extended version of the Defects4J dataset that consists of 15 Java projects with 594 known bugs. To decide which of these bugs the tools detect, we use a novel methodology that combines an automatic analysis of warnings and bugs with a manual validation of each candidate of a detected bug. The results of the study show that: (i) static bug detectors find a non-negligible amount of all bugs, (ii) different tools are mostly complementary to each other, and (iii) current bug detectors miss the large majority of the studied bugs. A detailed analysis of bugs missed by the static detectors shows that some bugs could have been found by variants of the existing detectors, while others are domain-specific problems that do not match any existing bug pattern. These findings help potential users of such tools to assess their utility, motivate and outline directions for future work on static bug detection, and provide a basis for future comparisons of static bug detection with other bug finding techniques, such as manual and automated testing.

International Conference on Automated Software Engineering (ASE '18), September 3–7, 2018, Montpellier, France. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3238147.3238213

1 INTRODUCTION

Finding software bugs is an important but difficult task. For average industry code, the number of bugs per 1,000 lines of code has been estimated to range between 0.5 and 25 [21]. Even after years of deployment, software still contains unnoticed bugs. For example, studies of the Linux kernel show that the average bug remains in the kernel for a surprisingly long period of 1.5 to 1.8 years [8, 24]. Unfortunately, a single bug can cause serious harm, even if it has been subsisting for a long time without doing so, as evidenced by examples of software bugs that have caused huge economic loses and even killed people [17, 28, 46].

Given the importance of finding software bugs, developers rely on several approaches to reveal programming mistakes. One approach is to identify bugs during the development process, e.g., through pair programming or code review. Another direction is testing, ranging from purely manual testing over semi-automated testing, e.g., via manually written but automatically executed unit tests, to fully automated testing, e.g., with UI-level testing tools. Once the software is deployed, runtime monitoring can reveal so far missed bugs, e.g., collect information about abnormal runtime



Tool	Bugs
Error Prone	8
Infer	5
SpotBugs	18
Total:	**31**
Total of 27 unique bugs	

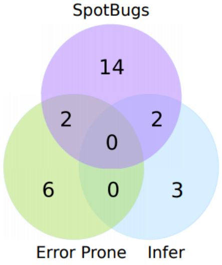


Figure 4: Total number of bugs found by all three static checkers and their overlap.

https://software-lab.org/publications/ase2018_static_bug_detectors_study.pdf

Summary

- Linters are cheap, fast, but imprecise analysis tools
  - Can be used for purposes other than bug detection (e.g., style)
- Conservative analyzers can demonstrate the absence of particular defects
  - At the cost of false positives due to necessary approximations
  - Inevitable trade-off between false positives and false negatives
- The best QA strategy involves multiple analysis and testing techniques
  - The exact set of tools and techniques depends on context