QA: Static Analysis

17-313 Fall 2025

Foundations of Software Engineering

https://cmu-17313q.github.io

Eduardo Feo Flushing

geek & poke
AT LEAST WE DON'T NEED TO OBFUSCATE IT BEFORE SHIPPING
RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE

Learning Goals

- Learn to get early feedback to reduce risk
- Find ways to catch our technical errors
- Gain an understanding of the relative strengths and weaknesses of static analysis
- Examine several popular analysis tools and understand their use cases
- Understand how analysis tools are used in large open source software

Administrivia

- Past Exams posted
- Cheat Sheet
  - One double-sided A4 .
  - You must submit it.
  - Handwritten = Bonus points.
  - Printed cheat sheets permitted but not awarded points.
- Midterm Next Wednesday, October 8th
- Review Session: Sunday during Recitation

P2B Grading Retrospective

- Improve Git usage
  - PRs not linked to issues
  - No dependencies / tags
- Align project board with repo
- Inconsistent PR quality (some good, some bad)
- Make your contributions visible

Smoking Section

- Last **two** full rows

Outline

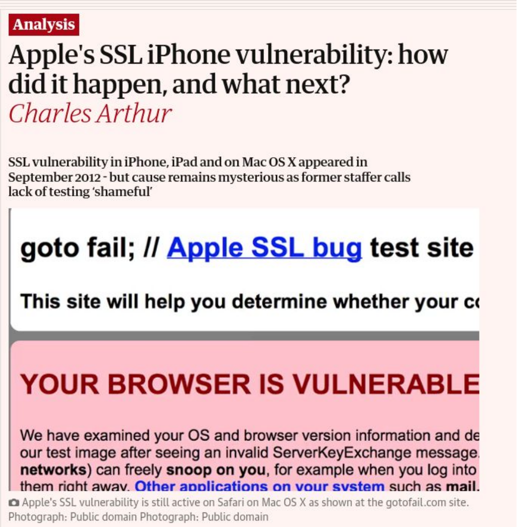
- **`goto fail;`** and similar unfamous bugs
- Static analysis tools
  - Linters for maintainability
  - Pattern-based static analyzers
- Challenges of static analysis

Outline

- **`goto fail;`** and similar unfamous bugs
- Static analysis tools
  - Linters for maintainability
  - Pattern-based static analyzers
- Challenges of static analysis

```c
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa,
                                 SSLBuffer signedParams,
                                 uint8_t *signature,
                                 UInt16 signatureLen) {
  OSStatus err;
   ....
  if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
  if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
  …
fail:
  SSLFreeBuffer(&signedHashes);
  SSLFreeBuffer(&hashCtx);
  return err;
}
```

goto fail;



Analysis

Apple's SSL iPhone vulnerability: how did it happen, and what next?

Charles Arthur

SSL vulnerability in iPhone, iPad and on Mac OS X appeared in September 2012 - but cause remains mysterious as former staffer calls lack of testing 'shameful'

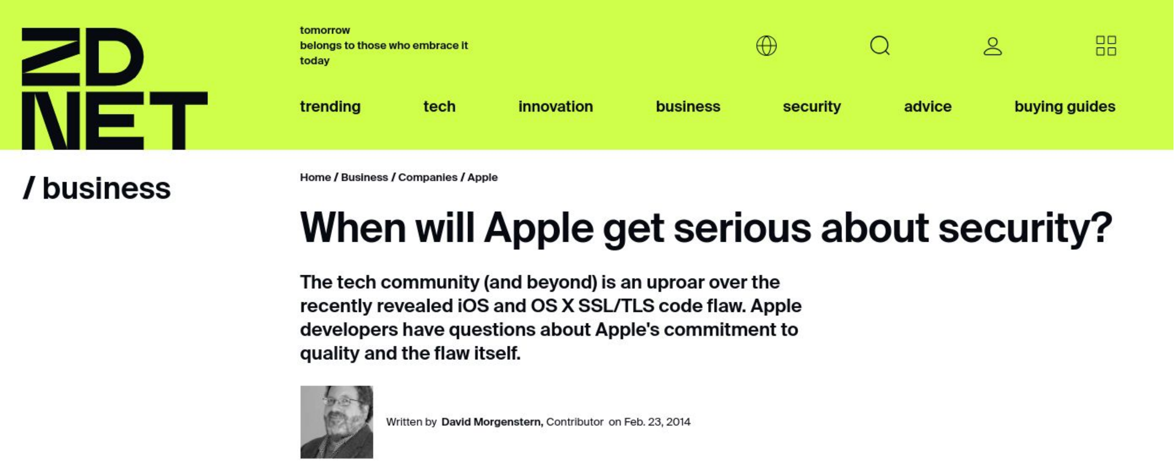
goto fail; // Apple SSL bug test site

This site will help you determine whether your c

YOUR BROWSER IS VULNERABLE

We have examined your OS and browser version information and de
our test image after seeing an invalid ServerKeyExchange message.
networks) can freely **snoop on you**, for example when you log into
them right away. Other applications on your system such as **mail**.

Apple's SSL vulnerability is still active on Safari on Mac OS X as shown at the gotofail.com site. Photograph: Public domain Photograph: Public domain



Home / Business / Companies / Apple

When will Apple get serious about security?

The tech community (and beyond) is an uproar over the recently revealed iOS and OS X SSL/TLS code flaw. Apple developers have questions about Apple's commitment to quality and the flaw itself.

Written by **David Morgenstern,** Contributor on Feb. 23, 2014

```c
/* from Linux 2.3.99 drivers/block/raid5.c */
static struct buffer_head *
get_free_buffer(struct stripe_head * sh,
                int b_size) {
  struct buffer_head *bh;
  unsigned long flags;
  save_flags(flags);
  cli(); // disables interrupts
  if ((bh = sh->buffer_pool) == NULL)
    return NULL;
  sh->buffer_pool = bh -> b_next;
  bh->b_size = b_size;
  restore_flags(flags); // re-enables interrupts
  return bh;
}
```

ERROR: function returns with interrupts disabled!

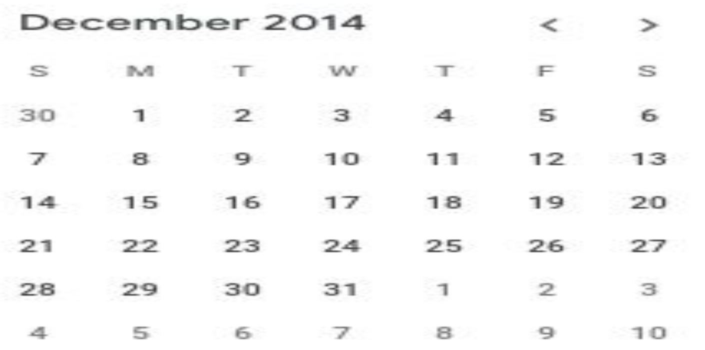
Twitter's week year bug

ISO 8601 rule: *The first week of the year is the week containing the first Thursday.*

If January 1 is a Thursday, then all days of that week (Mon Dec 29 – Sun Jan 4) belong to week 01 of the new year.

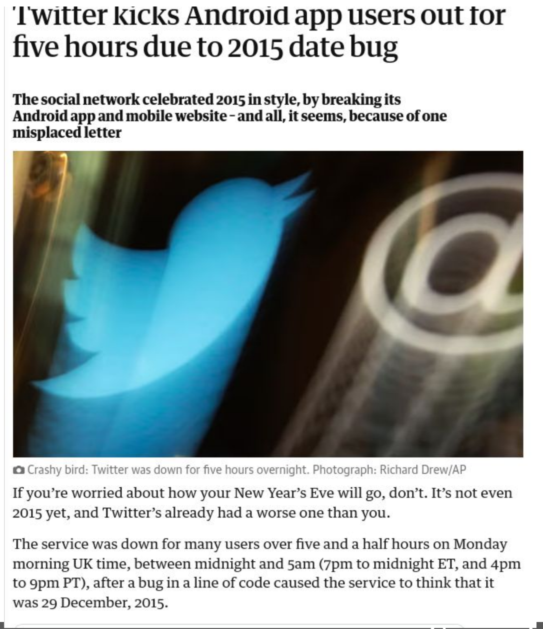
```
DateTimeFormatter.ofPattern("dd MMM YYYY").format(zonedDateTime)
```

Use yyyy instead of YYYY



December 2014 < >

S	M	T	W	T	F	S
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10



Twitter kicks Android app users out for five hours due to 2015 date bug

The social network celebrated 2015 in style, by breaking its Android app and mobile website - and all, it seems, because of one misplaced letter

Crashy bird: Twitter was down for five hours overnight. Photograph: Richard Drew/AP

If you're worried about how your New Year's Eve will go, don't. It's not even 2015 yet, and Twitter's already had a worse one than you.

The service was down for many users over five and a half hours on Monday morning UK time, between midnight and 5am (7pm to midnight ET, and 4pm to 9pm PT), after a bug in a line of code caused the service to think that it was 29 December, 2015.

Could you have found them?

- How often would those bugs trigger?
- Driver bug:
  - What happens if you return from a driver with interrupts disabled?
  - Consider: that’s one function
    - …in a 2000 LOC file
    - …in a module with 60,000 LOC
    - …IN THE LINUX KERNEL

Some defects are very difficult to find via testing, inspection.

Defects of interest…

- Are on uncommon or difficult-to-force execution paths. (vs testing)
- Executing (or interpreting/otherwise analyzing) all paths concretely to find such defects is infeasible.
- What we really want to do is check the **entire possible state space** of the program for particular properties.
- What we **CAN** do is check an **abstract state space** of the program for particular properties.

Activity: Analyze the Python program statically

```
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 is Static Analysis?

- **Systematic** examination of an **abstraction** of program **state space**.
    - Does not execute code! (like code review)
- **Abstraction:** produce a representation of a program that is simpler to analyze.
    - Results in fewer states to explore; makes difficult problems tractable.
- Check if a **particular property** holds over the entire state space:
- Liveness: “something good eventually happens.”
    - Safety: “this bad thing can’t ever happen.”
    - Compliance with mechanical design rules.

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` ``?

What static analysis can and cannot do

- 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

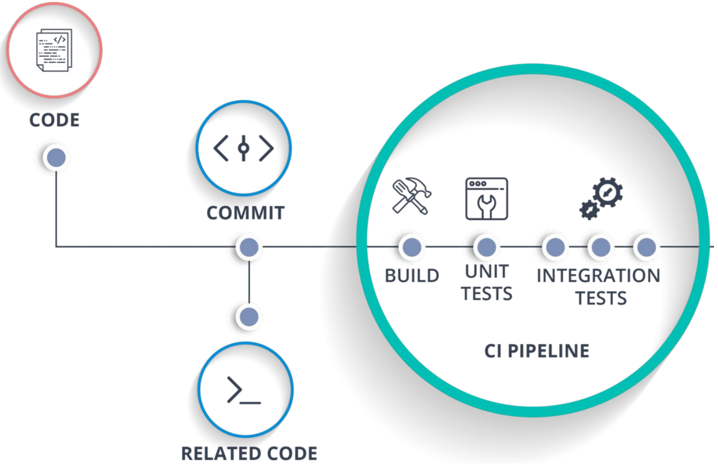
Outline

- `goto fail;` and similar unfamous bugs
- **Static analysis tools**
  - Linters for maintainability
  - Pattern-based static analyzers
- Challenges of static analysis

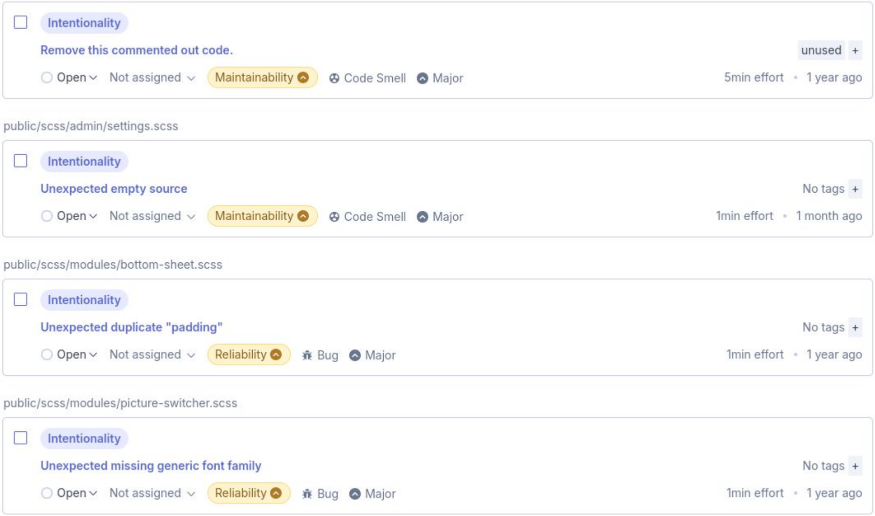
Tools for Static Analysis

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

Static analysis is a key part of CI



GitHub Actions



□ Intentionality

Remove this commented out code.

unused +

○ Open ⌵ Not assigned ⌵ Maintainability Code Smell Major 5min effort • 1 year ago

public/scss/admin/settings.scss

□ Intentionality

Unexpected empty source

No tags +

○ Open ⌵ Not assigned ⌵ Maintainability Code Smell Major 1min effort • 1 month ago

public/scss/modules/bottom-sheet.scss

□ Intentionality

Unexpected duplicate "padding"

No tags +

○ Open ⌵ Not assigned ⌵ Reliability Bug Major 1min effort • 1 year ago

public/scss/modules/picture-switcher.scss

□ Intentionality

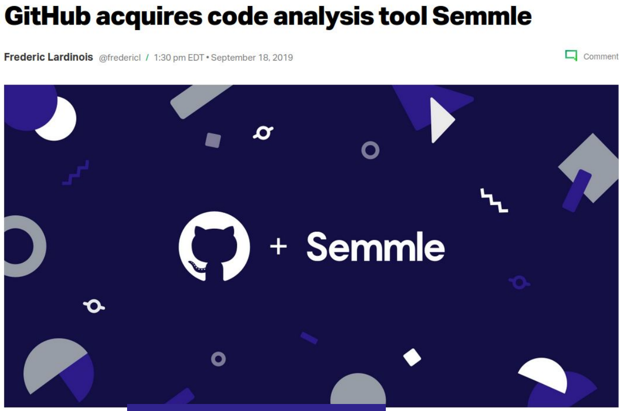
Unexpected missing generic font family

No tags +

○ Open ⌵ Not assigned ⌵ Reliability Bug Major 1min effort • 1 year ago

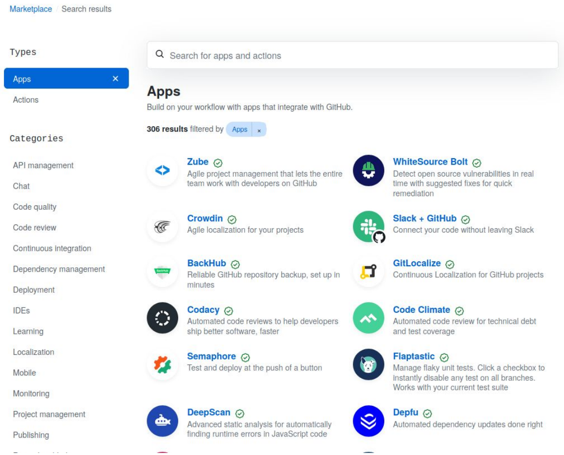
sonarcloud

Static analysis used to be an academic amusement; now it's heavily commercialized



GitHub acquires code analysis tool Semmle

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



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

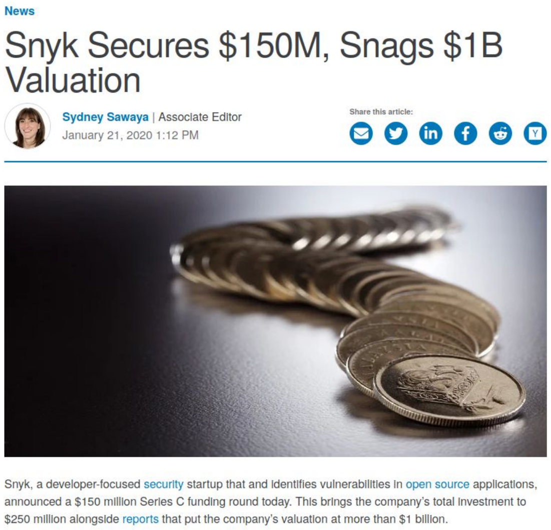
Semaphore Test and deploy at the push of a button

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

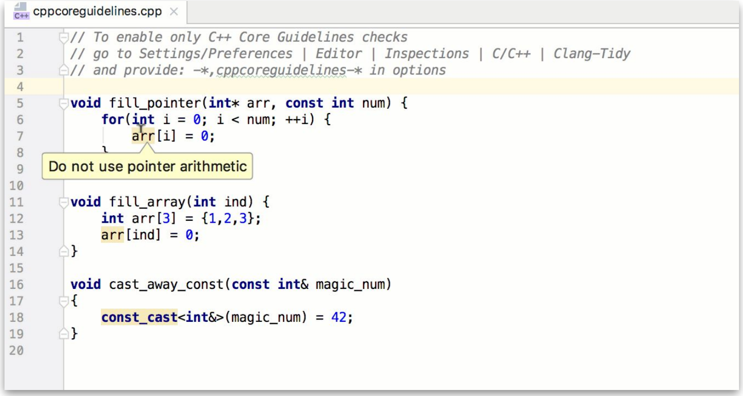
Sydney Sawaya | Associate Editor
January 21, 2020 1:12 PM

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

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

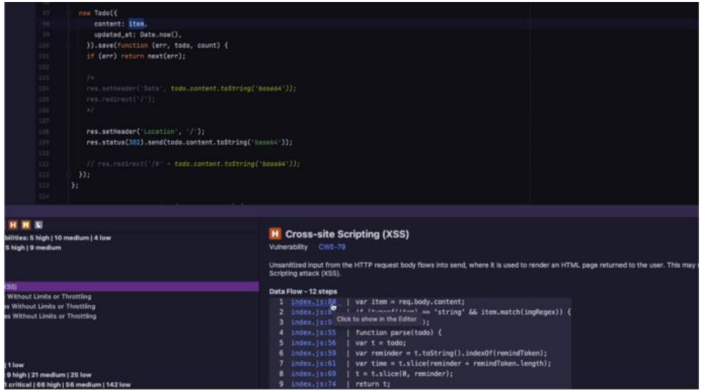
Static analysis is also integrated into IDEs



cppcoreguidelines.cpp

```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



```javascript
  new Todo({
    content: item,
    updated_at: Date.now(),
  }).save(function (err, todo, count) {
  if (err) return next(err);

  /*
  res.setHeader('Data', todo.content.toString('base64'));
  res.redirect('/');
  */

  res.setHeader('Location', '/');
  res.status(302).send(todo.content.toString('base64'));

  // res.redirect('/#' + todo.content.toString('base64'));
  });
};
```

bilities: 5 high | 10 medium | 4 low
5 high | 9 medium

XSS)
Without Limits or Throttling
s Without Limits or Throttling
es Without Limits or Throttling

| 1 low
9 high | 21 medium | 25 low
critical | 66 high | 56 medium | 142 low

H Cross-site Scripting (XSS)

Vulnerability CWE-79

Unsanitized input from the HTTP request body flows into send, where it is used to render an HTML page returned to the user. This may
Scripting attack (XSS).

Data Flow - 12 steps

```
1 index.js:80  | var item = req.body.content;
2 index.js:8   | if (typeof(item) == 'string' && item.match(imgRegex)) {
3 index.js:9                                   );
4 index.js:55  | function parse(todo) {
5 index.js:56  | var t = todo;
6 index.js:59  | var reminder = t.toString().indexOf(remindToken);
7 index.js:61  | var time = t.slice(reminder + remindToken.length);
8 index.js:69  | t = t.slice(0, reminder);
9 index.js:74  | return t;
```

Linters

Cheap, fast, and lightweight static source analysis

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

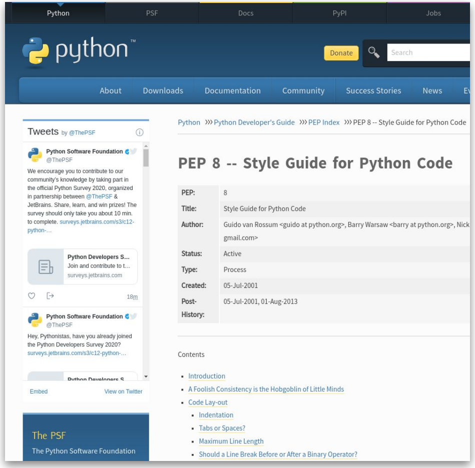
Use linters to improve maintainability

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

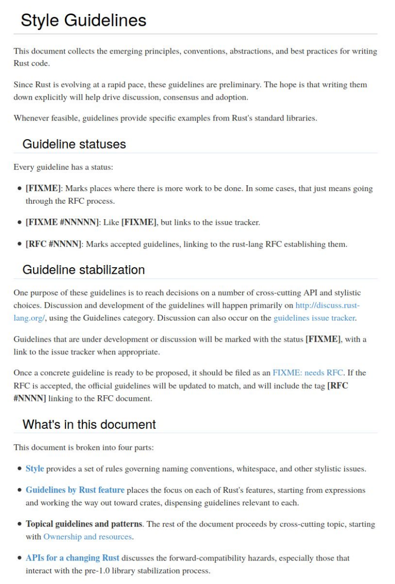
Python ››› Python Developer's Guide ››› PEP Index ››› PEP 8 -- Style Guide for Python Code

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.

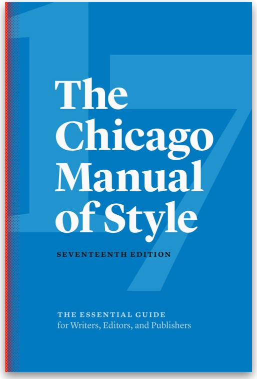
Guidelines that are under development or discussion will be marked with the status **[FIXME]**, with a link to the issue tracker when appropriate.

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

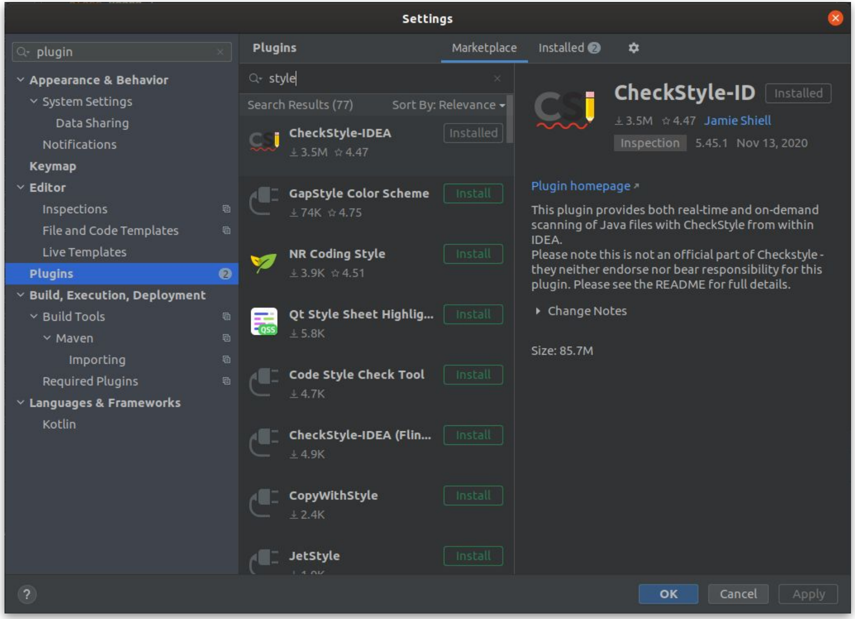
Use linters to enforce style guidelines

Don’t rely on manual inspection during code review!

CSI
RuboCop
Black
Java™
python™

Automatically reformat your existing code

Developer time is valuable!



style

Search Results (77) Sort By: Relevance

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

▸ Change Notes

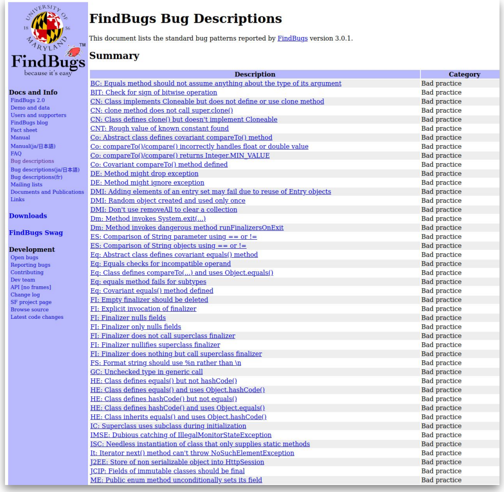
Size: 85.7M

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

Pattern-based Static Analysis Tools

- 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

SpotBugs

http://findbugs.sourceforge.net/bugDescriptions.html

https://spotbugs.readt

Example: Bad Practice

```
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!");
}
```

ES_COMPARING_STRINGS_WITH_EQ

Comparing strings with ==

Example: Bad Practice

```
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!");
}
```

ES_COMPARING_STRINGS_WITH_EQ

Comparing strings with ==

Example: Performance

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

SBSC_USE_STRINGBUFFER_CONCATENATION

Method concatenates strings using + in a loop

Example: Performance

```
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();
}
```

SBSC_USE_STRINGBUFFER_CONCATENATION

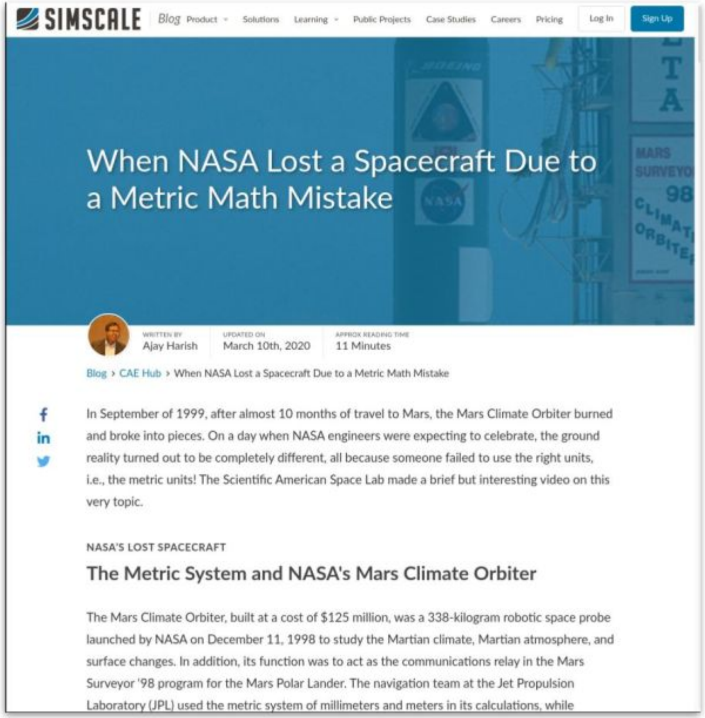
Method concatenates strings using + in a loop

Use type annotations to detect common errors

- Uses a conservative analysis to prove the absence of certain defects:
    - Unsanitized input, Null pointer errors, uninitialized fields, certain liveness issues, information leaks, SQL injections, bad regular expressions, incorrect physical units, bad format strings, …
- Assuming that code is annotated and those annotations are correct
- Use annotations to enhance type system

AUTH
NASA
MARS ORBITER
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

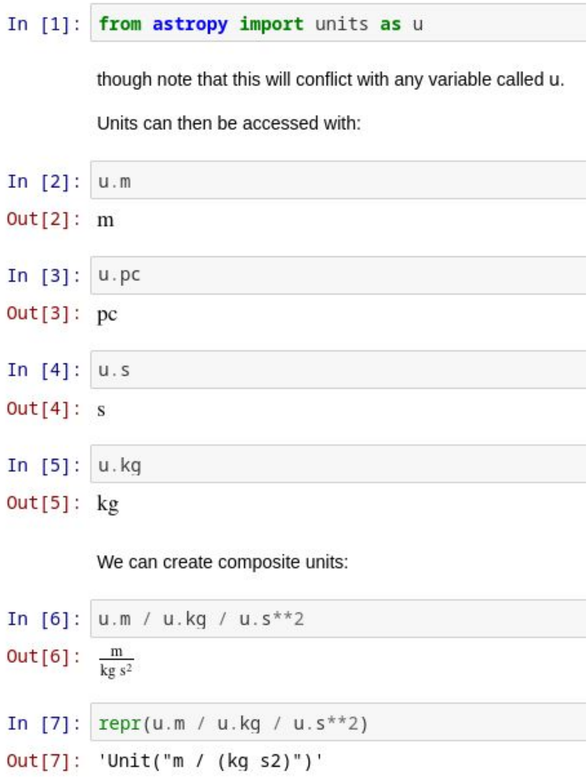
The Mars Climate Orbiter, built at a cost of $125 million, was a 338-kilogram robotic space probe launched by NASA on December 11, 1998 to study the Martian climate, Martian atmosphere, and surface changes. In addition, its function was to act as the communications relay in the Mars Surveyor '98 program 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

Does this program compile? **No**

```
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



```
In [1]: from astropy import units as u
```

though note that this will conflict with any variable called u.

Units can then be accessed with:

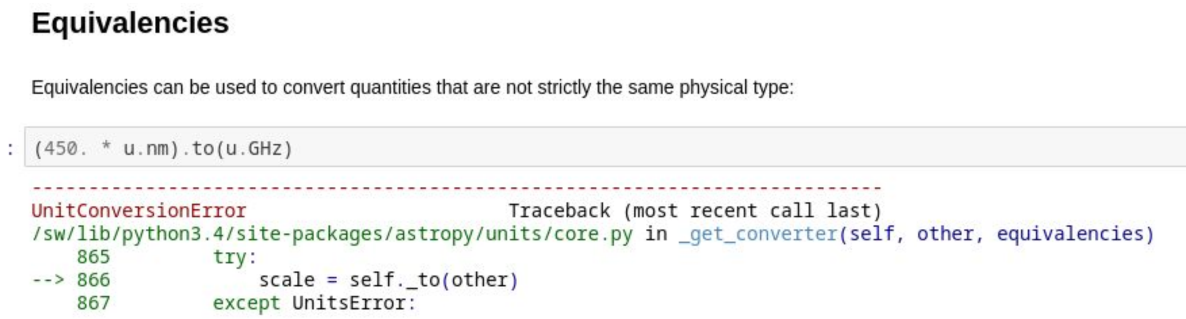
```
In [2]: u.m
Out[2]: m

In [3]: u.pc
Out[3]: pc

In [4]: u.s
Out[4]: s

In [5]: u.kg
Out[5]: kg
```

We can create composite units:

```
In [6]: u.m / u.kg / u.s**2
```

Out[6]: $\frac{\mathrm{m}}{\mathrm{kg\ s^{2}}}$

```
In [7]: repr(u.m / u.kg / u.s**2)
Out[7]: 'Unit("m / (kg s2)")'
```



Equivalencies

Equivalencies can be used to convert quantities that are not strictly the same physical type:

```
: (450. * u.nm).to(u.GHz)

---------------------------------------------------------------------------
UnitConversionError                       Traceback (most recent call last)
/sw/lib/python3.4/site-packages/astropy/units/core.py in _get_converter(self, other, equivalencies)
    865         try:
--> 866             scale = self._to(other)
    867         except UnitsError:
```

“Malicious” User Inputs

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

Taint Analysis

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

Taint Checking using Annotations

Indicates that data is tainted

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

Argument ***must*** be untainted

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

Guarantees that return value is untainted

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

Does this program compile? **No**

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

input is NOT guaranteed to be @Untainted

Does this program compile? **Yes**

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

Outline

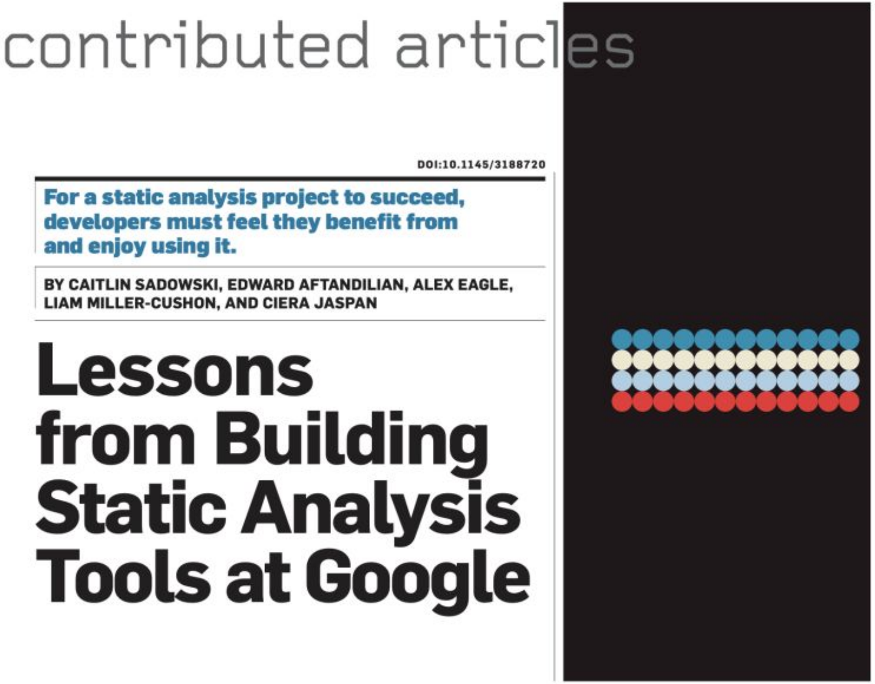
- **`goto fail;`** and similar unfamous bugs
- Static analysis tools
  - Linters for maintainability
  - Pattern-based static analyzers
- **Challenges of static analysis**

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

Lessons for Static Analysis Tools at Google

- Make It a Compiler Workflow
- Value of compiler checks.
- Reporting issues sooner is better
- Warn During Code Review
- Engineers working on static analysis must demonstrate impact through hard data.



contributed articles

DOI:10.1145/3188720

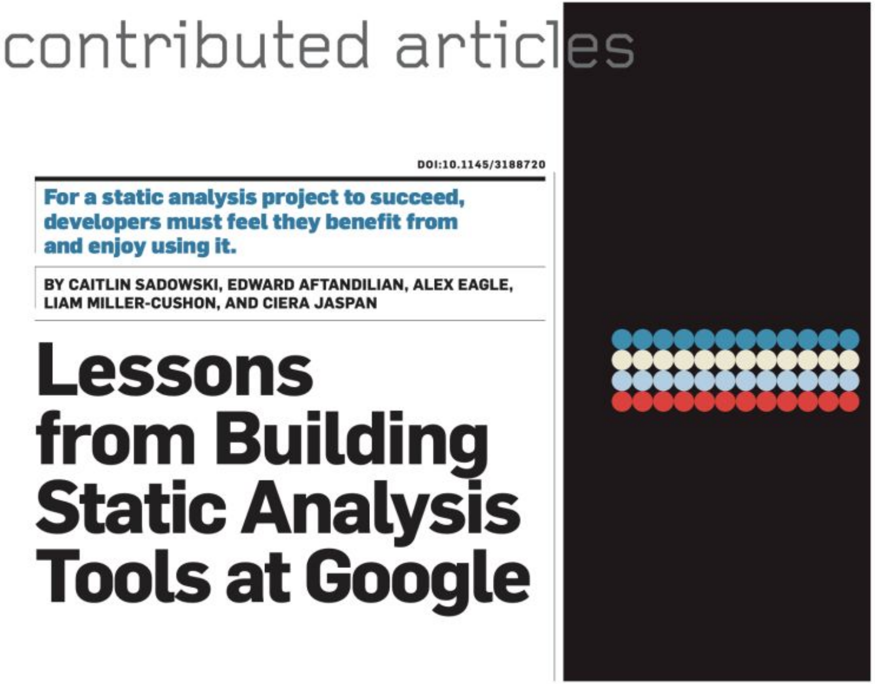
For a static analysis project to succeed, developers must feel they benefit from and enjoy using it.

BY CAITLIN SADOWSKI, EDWARD AFTANDILIAN, ALEX EAGLE, LIAM MILLER-CUSHON, AND CIERA JASPAN

Lessons from Building Static Analysis Tools at Google

Lessons for Static Analysis Tools at Google

- Finding bugs is easy
- Most developers will not go out of their way to use static analysis tools.
- Developer happiness is key.
- Do not just find bugs, fix
- them.
- Crowdsource analysis
- development.



contributed articles

DOI:10.1145/3188720

For a static analysis project to succeed, developers must feel they benefit from and enjoy using it.

BY CAITLIN SADOWSKI, EDWARD AFTANDILIAN, ALEX EAGLE, LIAM MILLER-CUSHON, AND CIERA JASPAN

Lessons from Building Static Analysis Tools at Google

Reasons engineers do not always use static analysis tools or ignore their warnings

- Not integrated.
  - The tool is not integrated into the developer's workflow or takes too long to run
- Not actionable
  - Whenever possible, the error should include a suggested fix that can be applied mechanically
- Not trustworthy
  - Users do not trust the results
- Not manifest in practice.
  - The reported bug is theoretically possible, but the problem does not actually manifest in practice
- Too expensive to fix.
  - Fixing the detected bug is too expensive or risky
- Warnings not understood

What you need to know

Early feedback through code reviews and static analysis is crucial for reducing risk and preventing technical errors.

Static analysis tools enhance code quality and maintainability while integrating seamlessly with CI for continuous checks.

Effective code reviews combine structured checklists with an empathetic, constructive approach to foster collaboration and improve code quality.

Static analysis has strengths in detecting issues like security vulnerabilities and performance problems, but it also has limitations and challenges.