Carnegie Mellon University

Static and Dynamic Analysis

17-313, Foundations of Software Engineering, Fall 2023

S3D Software and Societal Systems Department

Learning Goals

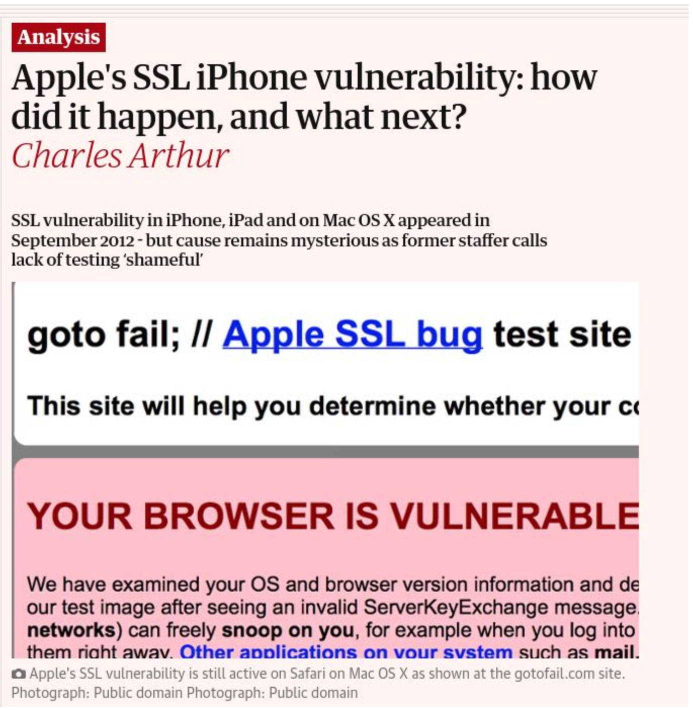
- Gain an understanding of the relative strengths and weaknesses of static and dynamic analysis
- Examine several popular analysis tools and understand their use cases
- Understand how analysis tools are used in large open source software

Outline

- `goto fail;` and similar unfamous bugs
- Static analysis vs dynamic analysis
- 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'

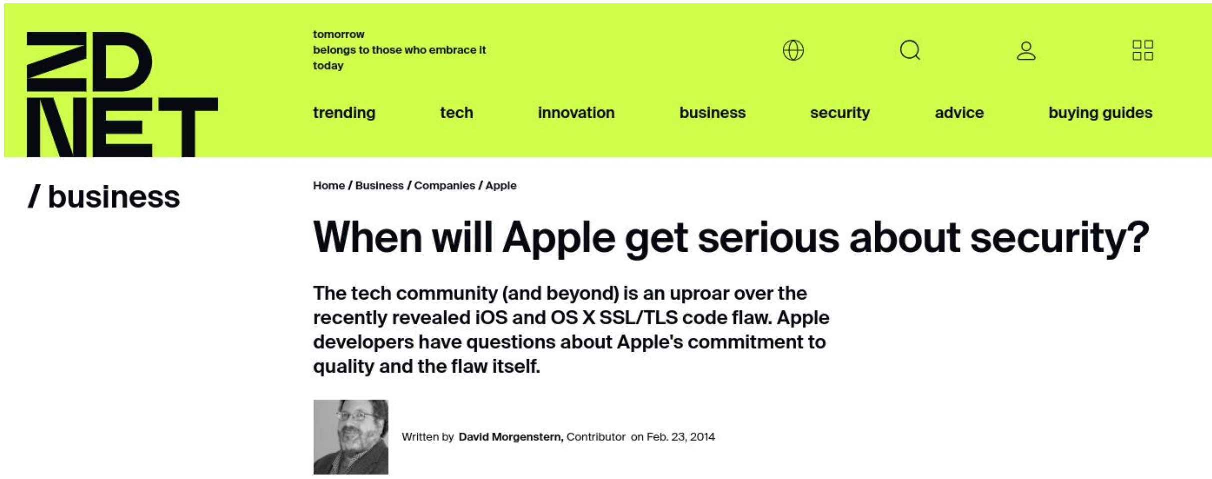
goto fail; // Apple SSL bug test site

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


ZDNET

tomorrow belongs to those who embrace it today

trending tech innovation business security advice buying guides

/ business

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;
}
```

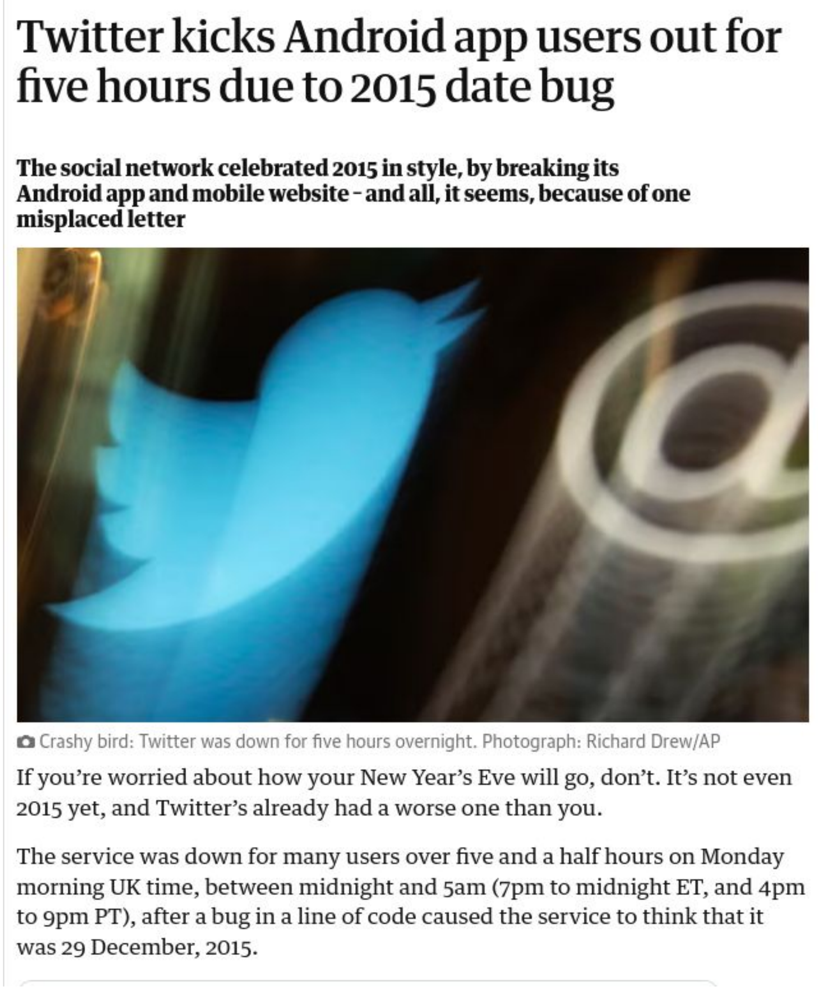
ERROR: function returns with interrupts disabled!

Twitter’s week year bug

ISO 8601 rule: *The first week of the year is the week containing the first Thursday.*
“So if January 1 falls on a Friday, it belongs to the last week of the previous year. If December 31 falls on a Wednesday, it belongs to week 01 of the following year.”

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

Use yyyy instead of YYYY



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

```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 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`?

- 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

- Requires only source code
- Conservatively reasons about all possible inputs and program paths
- Reported warnings may contain false positives
- Can report all warnings of a particular class of problems
- Advanced techniques like verification can prove certain complex properties, but rarely run in CI due to cost

Dynamic Analysis

- Requires successful build + test inputs
- Observes individual executions
- Reported problems are real, as observed by a witness input
- Can only report problems that are seen. Highly dependent on test inputs. Subject to false negatives
- 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 1856

FindBugs™
because it's easy

: my[py]

snyk

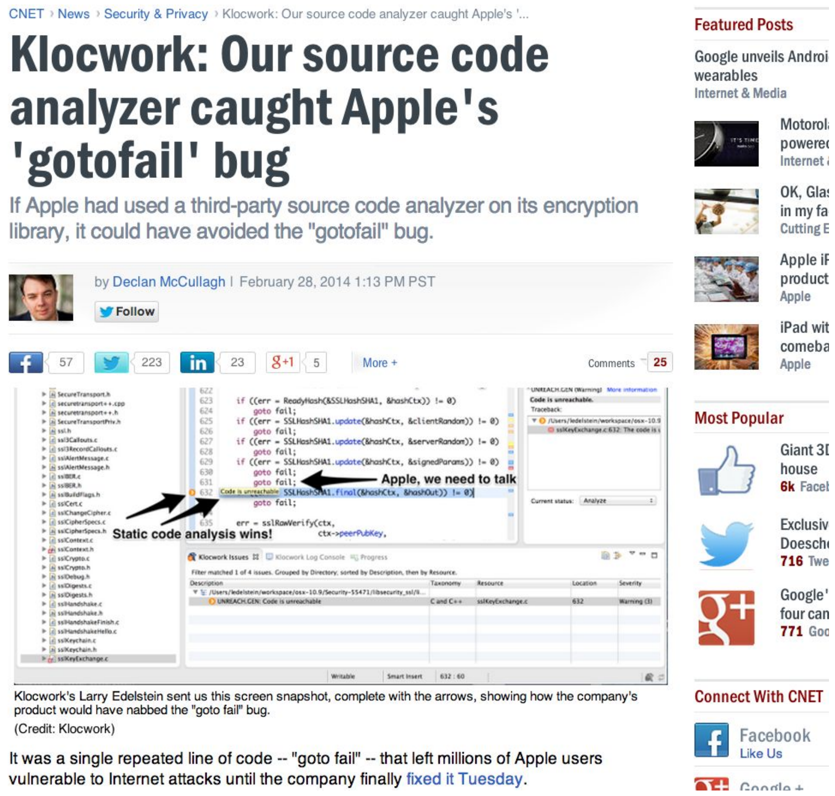
sonarqube

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



CNET › News › Security & Privacy › Klocwork: Our source code analyzer caught Apple's '...

Klocwork: Our source code analyzer caught Apple's 'gotofail' bug

If Apple had used a third-party source code analyzer on its encryption library, it could have avoided the "gotofail" bug.

by Declan McCullagh | February 28, 2014 1:13 PM PST

SecureTransport.h
securetransport++.cpp
securetransport++.h
SecureTransportPriv.h
ssl.h
ssl3Callouts.c
ssl3RecordCallouts.c
sslAlertMessage.c
sslAlertMessage.h
sslBER.c
sslBER.h
sslBuildFlags.h
sslCert.c
sslChangeCipher.c
sslCipherSpecs.c
sslCipherSpecs.h
sslContext.c
sslContext.h
sslCrypto.c
sslCrypto.h
sslDebug.h
sslDigests.c
sslDigests.h
sslHandshake.c
sslHandshake.h
sslHandshakeFinish.c
sslHandshakeHello.c
sslKeychain.c
sslKeychain.h
sslKeyExchange.c

```
622
623     if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
624         goto fail;
625     if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
626         goto fail;
627     if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
628         goto fail;
629     if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
630         goto fail;
631         goto fail;
632 Code is unreachable SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
            goto fail;

635     err = sslRawVerify(ctx,
                           ctx->peerPubKey,
```

UNREACH.GEN (Warning) More information
Code is unreachable.
Traceback:
/Users/ledelstein/workspace/osx-10.9
sslKeyExchange.c:632: The code is
Current status: Analyze

Filter matched 1 of 4 issues. Grouped by Directory; sorted by Description, then by Resource.

Description	Taxonomy	Resource	Location	Severity
/Users/ledelstein/workspace/osx-10.9/Security-55471/libsecurity_ssl/li...				
UNREACH.GEN: Code is unreachable	C and C++	sslKeyExchange.c	632	Warning (3)

Klocwork's Larry Edelstein sent us this screen snapshot, complete with the arrows, showing how the company's product would have nabbed the "goto fail" bug.
(Credit: Klocwork)

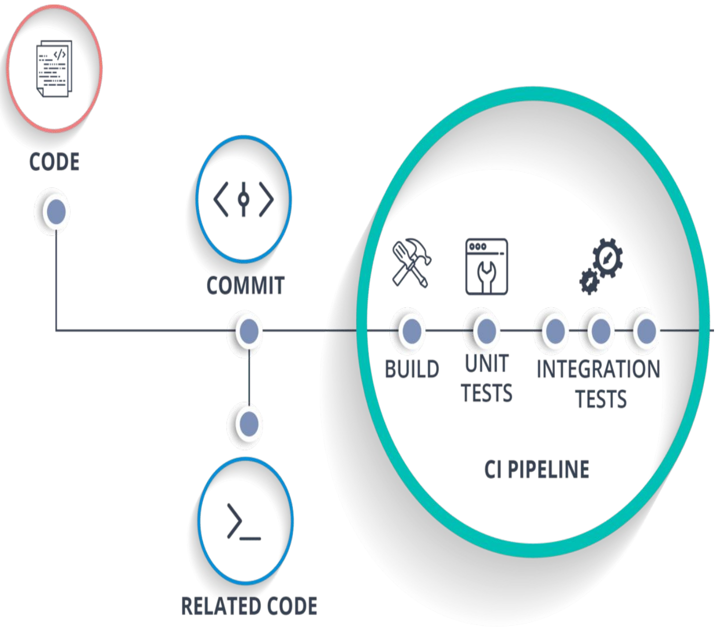
It was a single repeated line of code -- "goto fail" -- that left millions of Apple users vulnerable to Internet attacks until the company finally fixed it Tuesday.

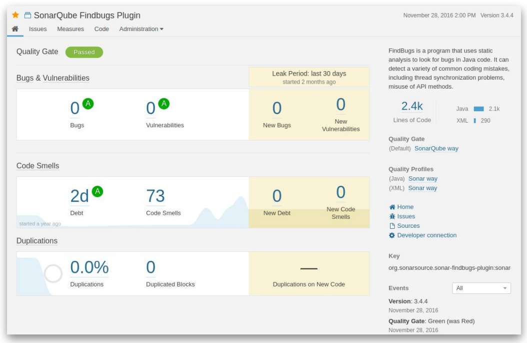
Featured Posts
Google unveils Androi
wearables
Internet & Media
Motorol
powere
Internet
OK, Glas
in my fa
Cutting E
Apple iF
product
Apple
iPad wit
comeba
Apple

Most Popular
Giant 3I
house
6k Facel
Exclusiv
Doesch
716 Twe
Google'
four can
771 Goo

Connect With CNET
Facebook
Like Us
Google +

Static analysis is a key part of continuous integration




SonarQube Findbugs Plugin

November 28, 2016 2:00 PM Version 3.4.4

Issues Measures Code Administration

Quality Gate Passed

Bugs & Vulnerabilities

Leak Period: last 30 days
started 2 months ago

0 A	0 A	0	0
Bugs	Vulnerabilities	New Bugs	New Vulnerabilities

Code Smells

2d A	73	0	0
Debt	Code Smells	New Debt	New Code Smells

started a year ago

Duplications

0.0%	0	—
Duplications	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

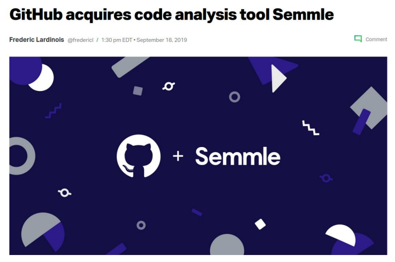
Travis CI

GitHub Actions

sonarqube

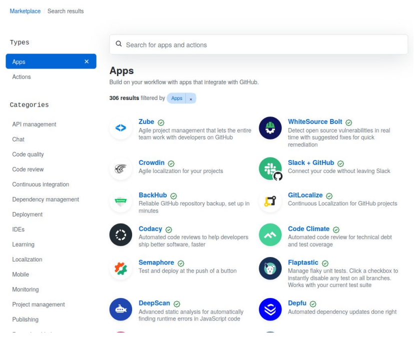
S3D

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

News

Snyk Secures $150M, Snags $1B Valuation

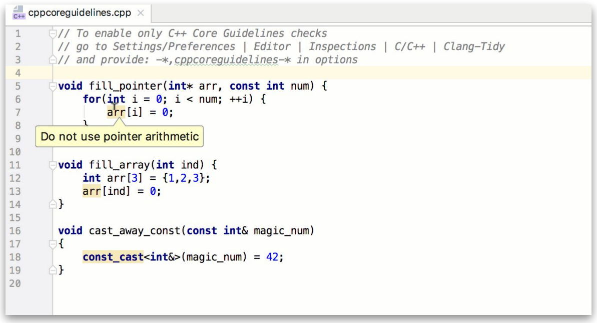
Sydney Sawaya | Associate Editor
January 21, 2020 1:12 PM

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.

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

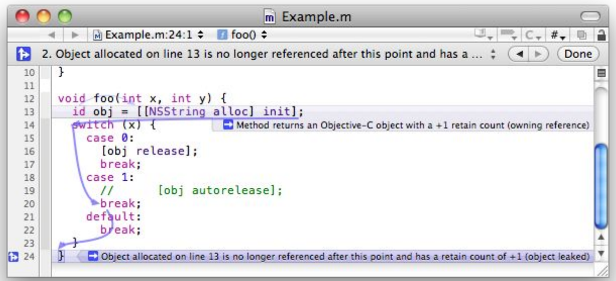
```
// To enable only C++ Core Guidelines checks
// go to Settings/Preferences | Editor | Inspections | C/C++ | Clang-Tidy
// and provide: -*,cppcoreguidelines-* in options

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

Do not use pointer arithmetic

```
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;
}
```



Example.m

Example.m:24:1 foo()

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

```
10  }
11
12  void foo(int x, int y) {
13    id obj = [[NSString alloc] init];
14    switch (x) {
15      case 0:
16        [obj release];
17        break;
18      case 1:
19        //       [obj autorelease];
20        break;
21      default:
22        break;
23    }
24  }
```

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)

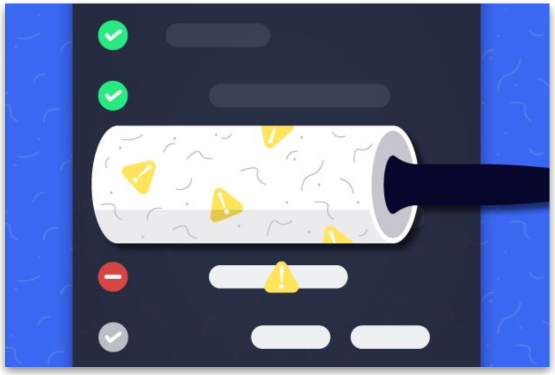
https://clang-analyzer.llvm.org

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
- …

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

Python Software Foundation @ThePSF

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

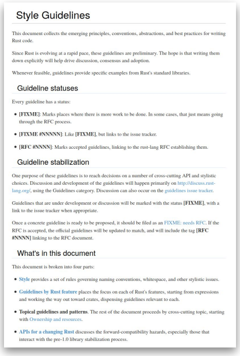
Python »» Python Developer's Guide »» PEP Index »» PEP 8 -- Style Guide for Python Code

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.

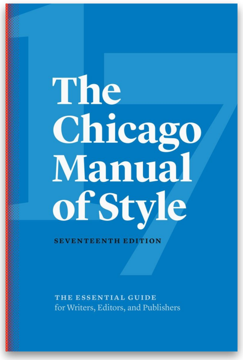
Guidelines that are under development or discussion will be marked with the status **[FIXME]**, with a link to the issue tracker when appropriate.

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

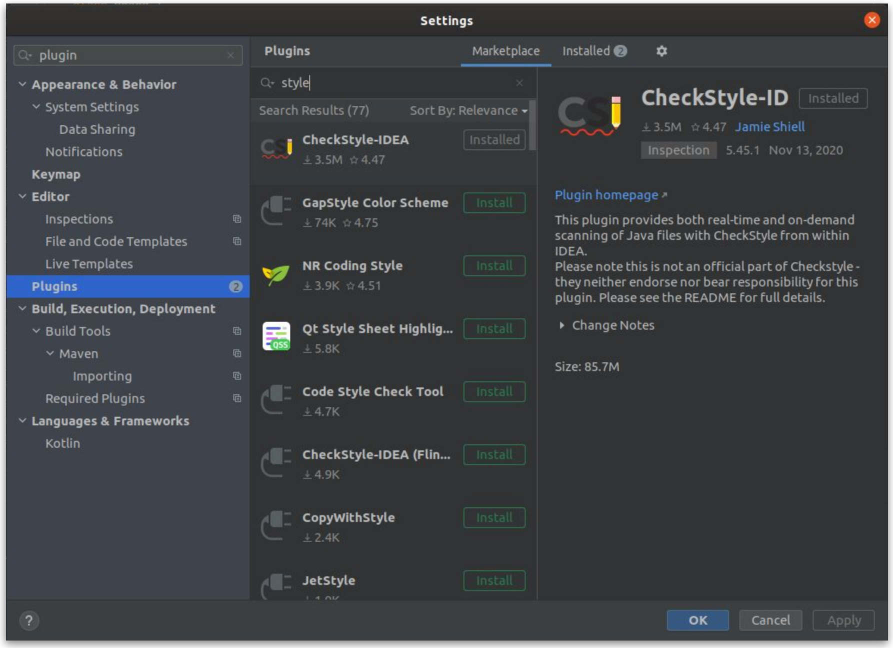
Use linters to enforce style guidelines

Don’t rely on manual inspection during code review!

https://checkstyle.sourceforge.io/

Automatically reformat your existing code

Developer time is valuable!


Settings

plugin

Appearance & Behavior
System Settings
Data Sharing
Notifications
Keymap
Editor
Inspections
File and Code Templates
Live Templates
Plugins 2
Build, Execution, Deployment
Build Tools
Maven
Importing
Required Plugins
Languages & Frameworks
Kotlin

Plugins — Marketplace — Installed 2

style

Search Results (77) Sort By: Relevance

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.

▸ Change Notes

Size: 85.7M

OK Cancel Apply

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

Pattern-Based Static Analyzers

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

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

University of Maryland 1856
FindBugs™
because it's easy

RuboCop

clang-tidy

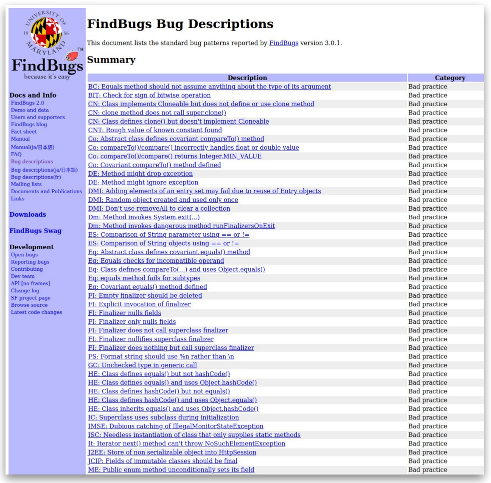
SpotBugs

Bandit

PMD
DON'T SHOOT THE MESSENGER

SpotBugs

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



FindBugs
because it's easy

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

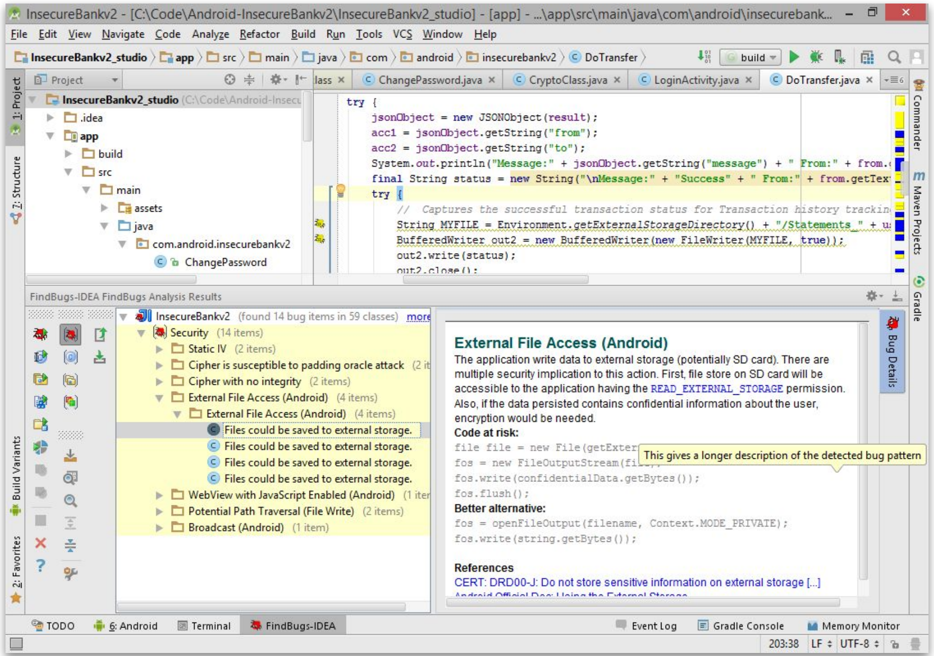
This document lists the standard bug patterns reported by FindBugs version 3.0.1.

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...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
InsecureBankv2_studio › app › src › main › java › com › android › insecurebankv2 › DoTransfer
build
Project
lass × ChangePassword.java × CryptoClass.java × LoginActivity.java × DoTransfer.java ×
InsecureBankv2_studio (C:\Code\Android-Insecu
.idea
app
build
src
main
assets
java
com.android.insecurebankv2
ChangePassword
1: Project
7: Structure
Build Variants
2: Favorites
Commander
Maven Projects
Gradle
Bug Details
TODO
6: Android
Terminal
FindBugs-IDEA
Event Log
Gradle Console
Memory Monitor
203:38 LF UTF-8

```
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 iter
    - 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();
```

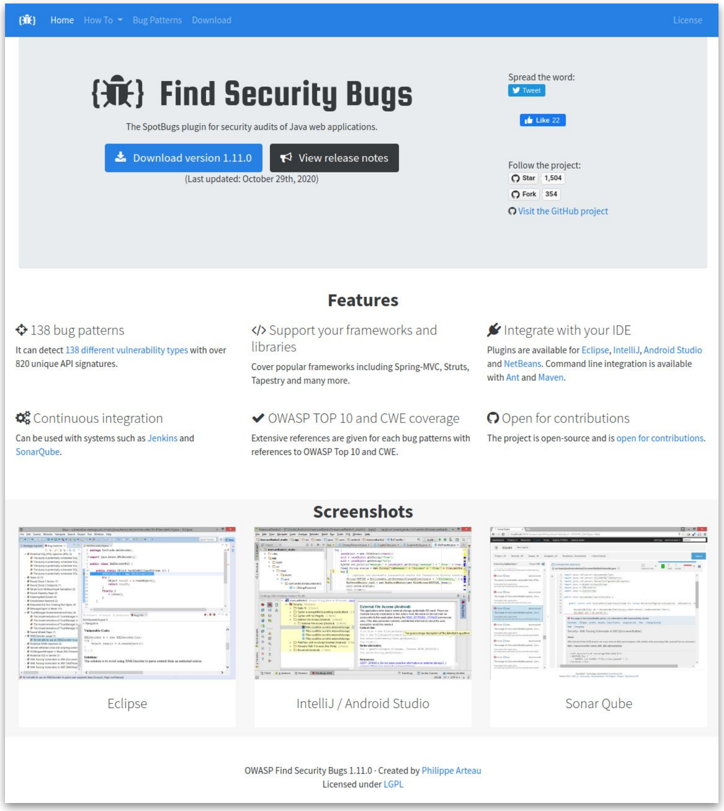
This gives a longer description of the detected bug pattern

Better alternative:

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

References

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



Find Security Bugs

The SpotBugs plugin for security audits of Java web applications.

(Last updated: October 29th, 2020)

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:

```
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

```
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

```
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

```
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();
}
```

Challenges of Static Analysis

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 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

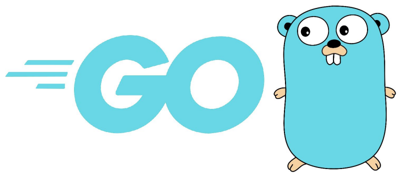
Error Prone

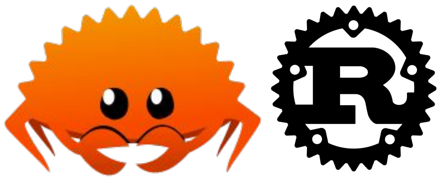
S3D

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

```
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.

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

Outline

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

Summary

- 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