Static Code Analysis using FindBugs Plugin

What is static analysis?

Analyzing code without executing it. Generally used to find bugs or ensure conformance to coding guidelines. The classic example is a compiler which finds lexical, syntactic and even some semantic mistakes.

When should you use it, and when shouldn't it be used?

Static analysis tools should be used when they help maintain code quality. If they're used, they should be integrated into the build process, otherwise they will be ignored.

***What are potential gotchas regarding proper and improper usage/application of static analysis*?**

Two common pathologies occur when using static analysis tools:

1. The tools produces spurious warnings/errors that the developers cannot silence. Eventually, most of the warnings are spurious and the developers stop paying attention to the output. This is why many teams require that code compile cleanly. If developers feel comfortable ignoring compiler warnings, the compile phase will eventually be filled with warning nobody ever pays attention to, even though they may be bugs.
2. The tools take too long to run and developers never bother to run them.

Any languages that don't have a good static analysis tool, and what do you do when you don't have an option for automated analysis?

For a number of reasons, many of the dynamic languages (ruby, python, perl) don't have static analysis tools that are as strong as those available in static languages. The standard method of finding bugs and making sure the code is working in dynamic languages are unit tests which help build confidence that the code actually works (hat-tip: Chris Conway).

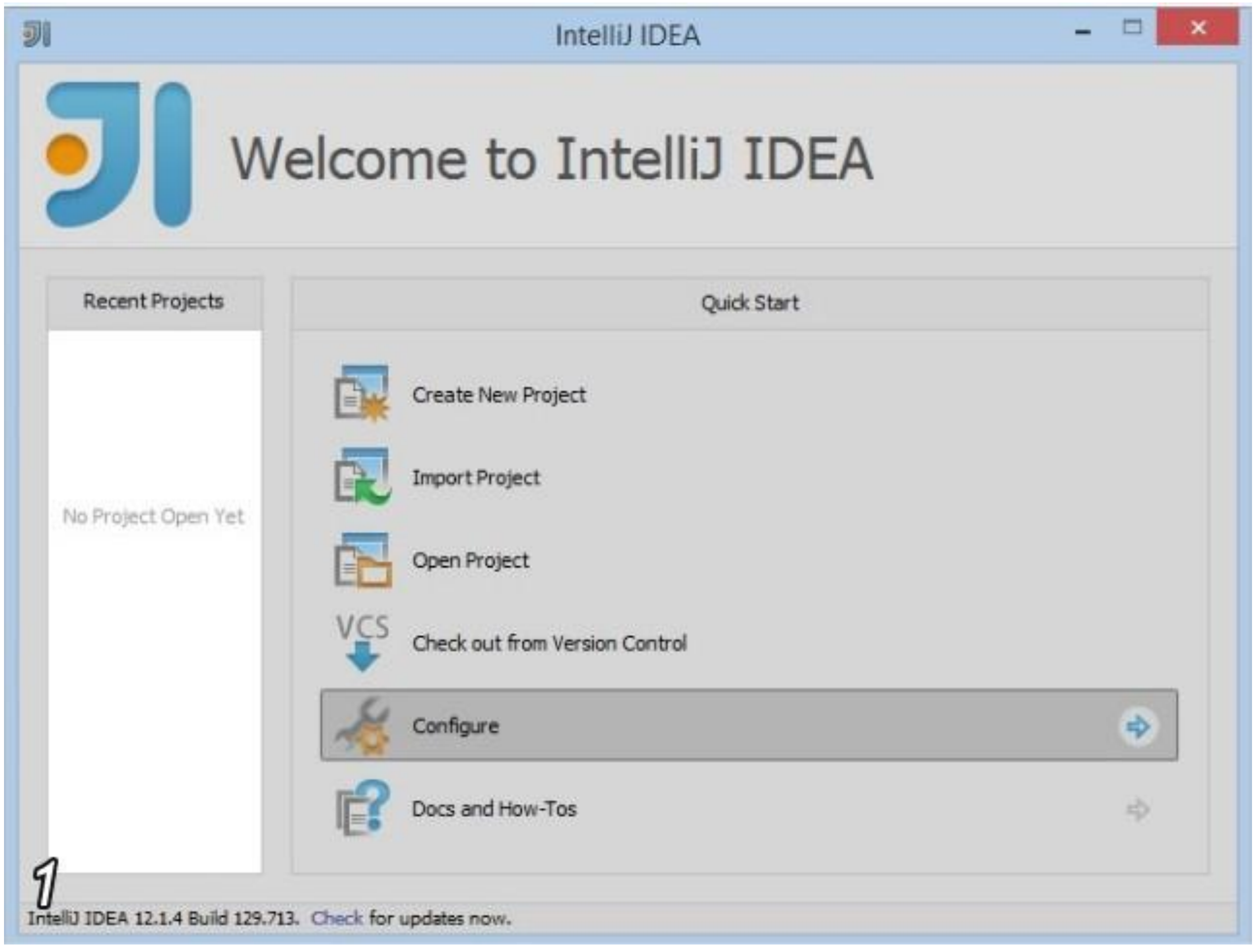
What is FindBugs Plugin?

FindBugs is an open source tool for static code analysis of Java programs. It scans byte code for so called *bug pattern* to find defects and/or suspicious code. Although FindBugs needs the compiled class files it is not necessary to execute the code for the analysis. Working with FindBugs helps to prevent from shipping avoidable issues. It is also an excellent motivation for improving the skills of development teams to write better code in the first place.

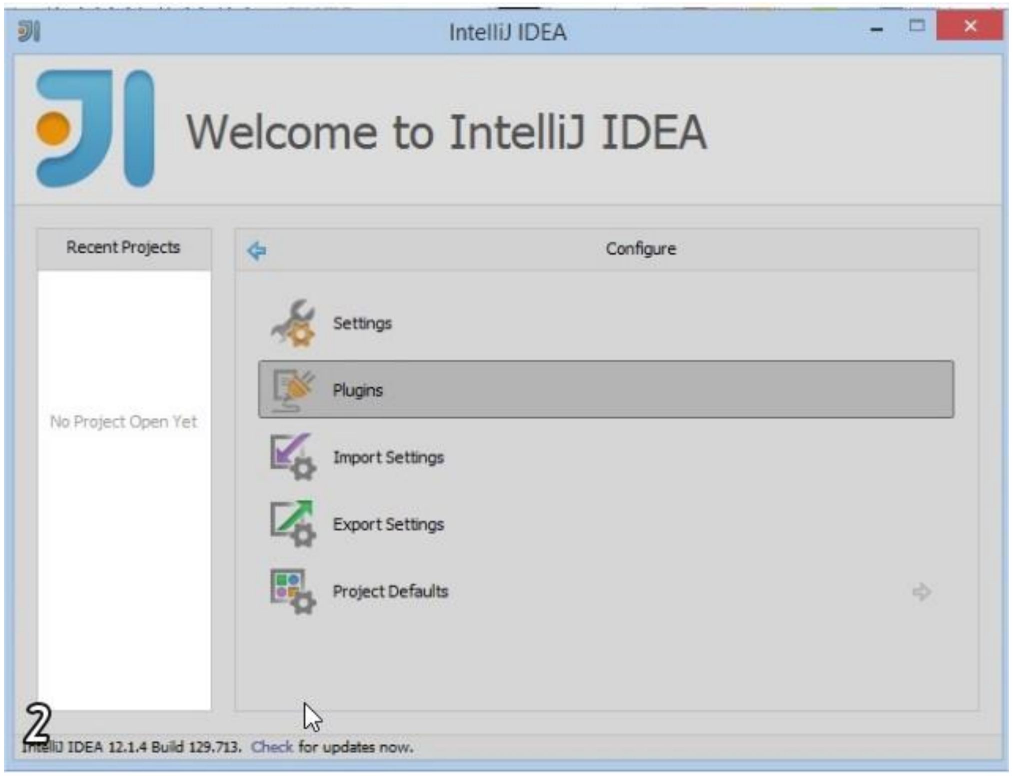
- Web Site: http://Findbugs.sourceforge.net/
- System requirements: Java 1.5 or higher
- License & Pricing: Open Source (Lesser GNU Public License)
- Support: Source Forge Project (http://sourceforge.net/projects/Findbugs/)

In this post I will be showing how to install FindBugs plugin in IntelliJ IDEA and perform a code analysis. I will also show how to use the FindSecurityBugs [3] plugin that comes within FindBugs for identifying the security weaknesses and bugs in your code.

Once you open IntelliJ IDEA, you can go to Configure -> Plugins in the opening window.



Next window we can see the configure menu, so now we need to go to the plugins. If you have already opened a project in IntelliJ IDEA, you can go to file -> Settings and in the left panel of the Settings window, select Plugins.

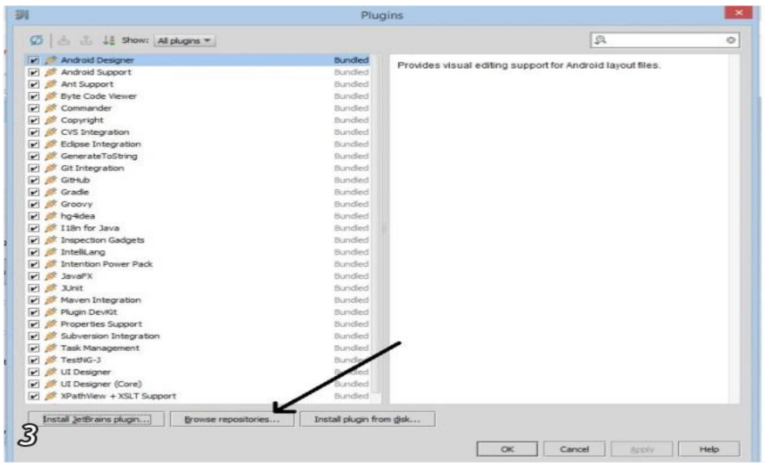


IntelliJ IDEA 12.1.4 Build 129.713. Check for updates now.

You can install the FindBugs plugin in two ways. If you have an internet connection, you can click on Browser repositories button and get the plugin installed. If not you can download the FindBugs plugin for IntelliJ IDEA and go with Install plugin from disk option where you can browse and provide the already downloaded plugin.

We can go through the Browser repositories button and get the plugin install and the second one is, if we can download the plugin, we can use Install plugin form disk button.

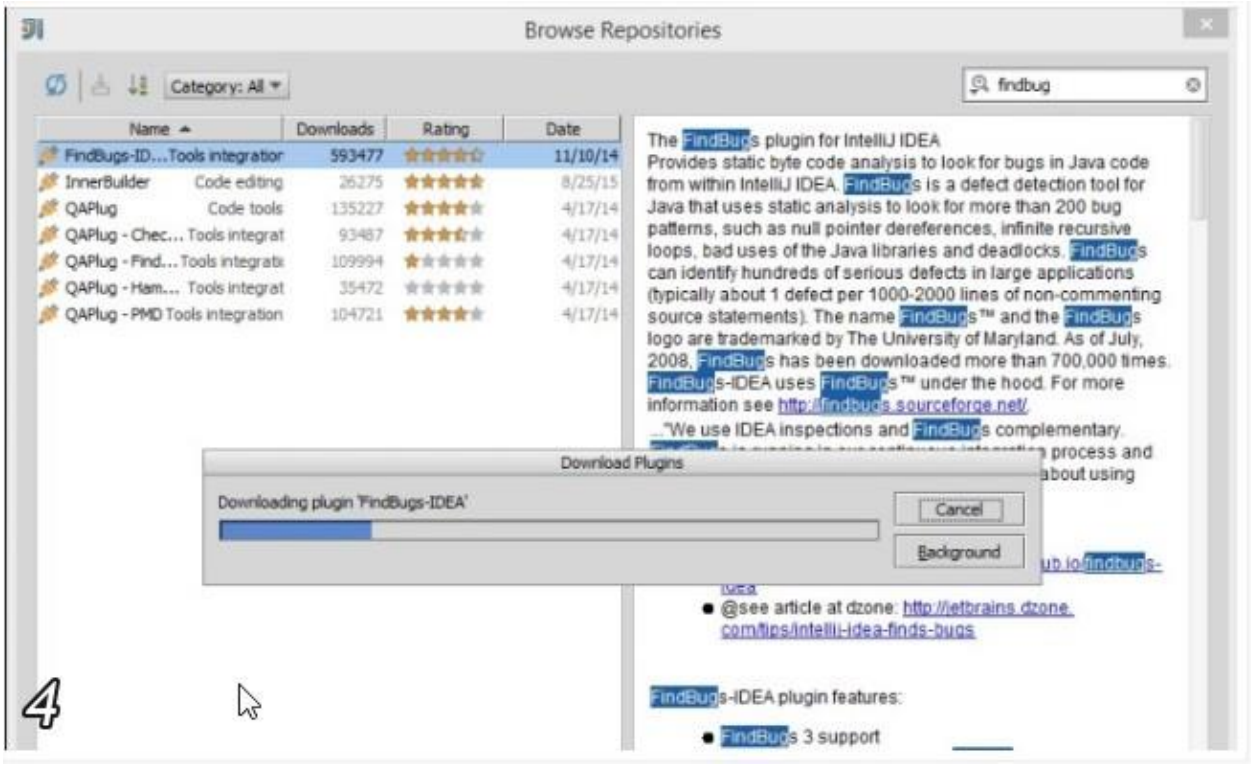
I used first option.



✓ Android Designer	Bundled
✓ Android Support	Bundled
✓ Ant Support	Bundled
✓ Byte Code Viewer	Bundled
✓ Commander	Bundled
✓ Copyright	Bundled
✓ CVS Integration	Bundled
✓ Eclipse Integration	Bundled
✓ GenerateToString	Bundled
✓ Git Integration	Bundled
✓ GitHub	Bundled
✓ Gradle	Bundled
✓ Groovy	Bundled
✓ hg4idea	Bundled
✓ I18n for Java	Bundled
✓ Inspection Gadgets	Bundled
✓ IntelliLang	Bundled
✓ Intention Power Pack	Bundled
✓ JavaFX	Bundled
✓ JUnit	Bundled
✓ Maven Integration	Bundled
✓ Plugin DevKit	Bundled
✓ Properties Support	Bundled
✓ Subversion Integration	Bundled
✓ Task Management	Bundled
✓ TestNG-J	Bundled
✓ UI Designer	Bundled
✓ UI Designer (Core)	Bundled
✓ XPathView + XSLT Support	Bundled

Provides visual editing support for Android layout files.

When you go with Browse repositories option, you can search for the FindBugs plugin and select FindBugs-IDEA and get it installed.



Name		Downloads	Rating	Date
FindBugs-ID…Tools integration		593477	★★★★☆	11/10/14
InnerBuilder	Code editing	26275	★★★★★	8/25/15
QAPlug	Code tools	135227	★★★★☆	4/17/14
QAPlug - Chec…	Tools integrat	93487	★★★☆☆	4/17/14
QAPlug - Find…	Tools integrati	109994	★☆☆☆☆	4/17/14
QAPlug - Ham…	Tools integrat	35472	☆☆☆☆☆	4/17/14
QAPlug - PMD	Tools integration	104721	★★★★☆	4/17/14

The FindBugs plugin for IntelliJ IDEA
Provides static byte code analysis to look for bugs in Java code from within IntelliJ IDEA. FindBugs is a defect detection tool for Java that uses static analysis to look for more than 200 bug patterns, such as null pointer dereferences, infinite recursive loops, bad uses of the Java libraries and deadlocks. FindBugs can identify hundreds of serious defects in large applications (typically about 1 defect per 1000-2000 lines of non-commenting source statements). The name FindBugs™ and the FindBugs logo are trademarked by The University of Maryland. As of July, 2008, FindBugs has been downloaded more than 700,000 times. FindBugs-IDEA uses FindBugs™ under the hood. For more information see http://findbugs.sourceforge.net/
..."We use IDEA inspections and FindBugs complementary.
process and
about using

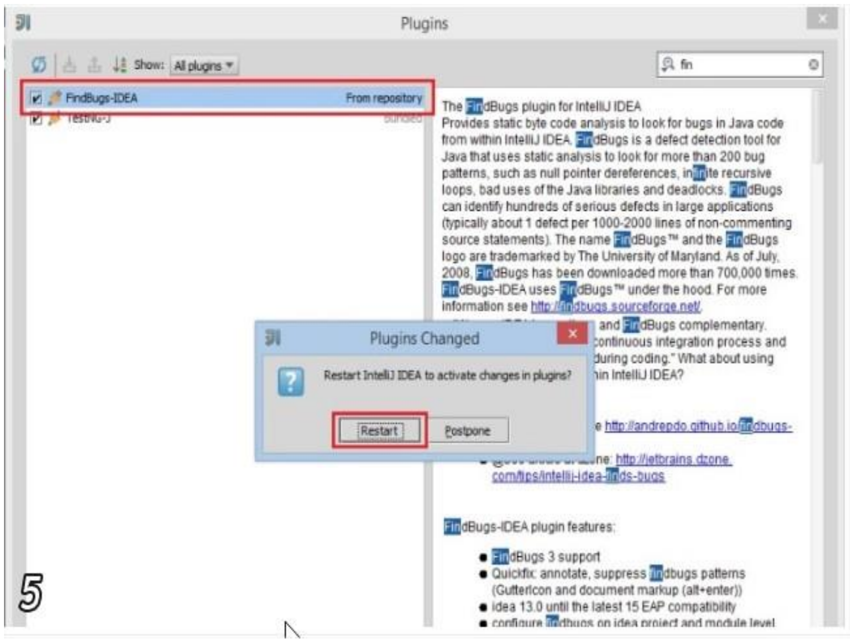
ub.io/findbugs-idea

- @see article at dzone: http://jetbrains.dzone.com/tips/intellij-idea-finds-bugs

FindBugs-IDEA plugin features:

- FindBugs 3 support

After complete the installation we need to restart the IDEA.



Restart IntelliJ IDEA to activate changes in plugins?

The FindBugs plugin for IntelliJ IDEA
Provides static byte code analysis to look for bugs in Java code from within IntelliJ IDEA. FindBugs is a defect detection tool for Java that uses static analysis to look for more than 200 bug patterns, such as null pointer dereferences, infinite recursive loops, bad uses of the Java libraries and deadlocks. FindBugs can identify hundreds of serious defects in large applications (typically about 1 defect per 1000-2000 lines of non-commenting source statements). The name FindBugs™ and the FindBugs logo are trademarked by The University of Maryland. As of July, 2008, FindBugs has been downloaded more than 700,000 times. FindBugs-IDEA uses FindBugs™ under the hood. For more information see http://findbugs.sourceforge.net/
and FindBugs complementary.
continuous integration process and
during coding." What about using
in IntelliJ IDEA?

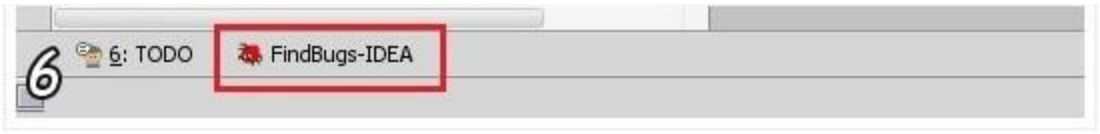
e http://andrepdo.github.io/findbugs-

- ne: http://jetbrains.dzone.com/tips/intellij-idea-finds-bugs

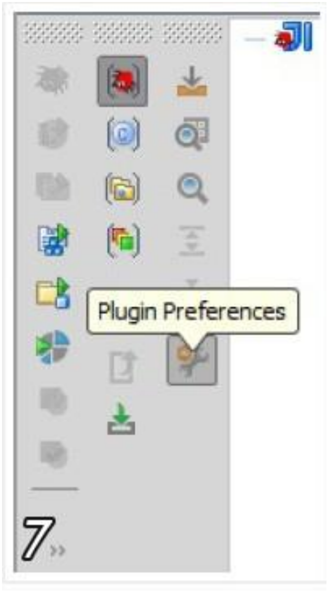
FindBugs-IDEA plugin features:

- FindBugs 3 support
- Quickfix: annotate, suppress findbugs patterns (GutterIcon and document markup (alt+enter))
- idea 13.0 until the latest 15 EAP compatibility
- configure findbugs on idea project and module level

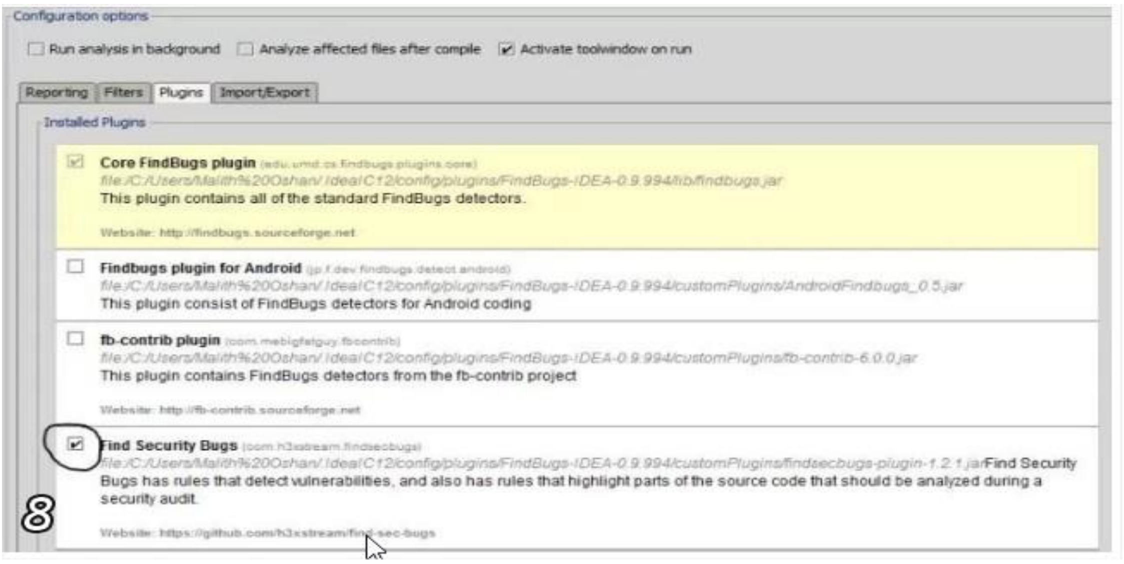
After restart the IDEA, in bottom of the IDEA we can see the FindBugs-IDEA button. Once you have installed the FindBugs plugin in IntelliJ IDEA, in the bottom of the IDE you will see the FindBugs-IDEA button. Upon clicking on it you can see all the settings of it in a panel.



Now we have to enable the FindSecurityBugs plugin which comes with FindBugs. This is for finding the security bugs in your code. Click on Plugin Preferences button.



Under the Plugins section of the General tab, click on the + button and select Add Find Security Bugs. Once the FindSecurityBugs plugin is added, click on apply and then OK.



Configuration options

☐ Run analysis in background ☐ Analyze affected files after compile ☑ Activate toolwindow on run

Reporting | Filters | Plugins | Import/Export

Installed Plugins

☑ **Core FindBugs plugin** (edu.umd.cs.findbugs.plugins.core)
file:/C:/Users/Malith%20Oshan/.IdeaIC12/config/plugins/FindBugs-IDEA-0.9.994/lib/findbugs.jar
This plugin contains all of the standard FindBugs detectors.
Website: http://findbugs.sourceforge.net

☐ **Findbugs plugin for Android** (jp.f.dev.findbugs.detect.android)
file:/C:/Users/Malith%20Oshan/.IdeaIC12/config/plugins/FindBugs-IDEA-0.9.994/customPlugins/AndroidFindbugs_0.5.jar
This plugin consist of FindBugs detectors for Android coding

☐ **fb-contrib plugin** (com.mebigfatguy.fbcontrib)
file:/C:/Users/Malith%20Oshan/.IdeaIC12/config/plugins/FindBugs-IDEA-0.9.994/customPlugins/fb-contrib-6.0.0.jar
This plugin contains FindBugs detectors from the fb-contrib project
Website: http://fb-contrib.sourceforge.net

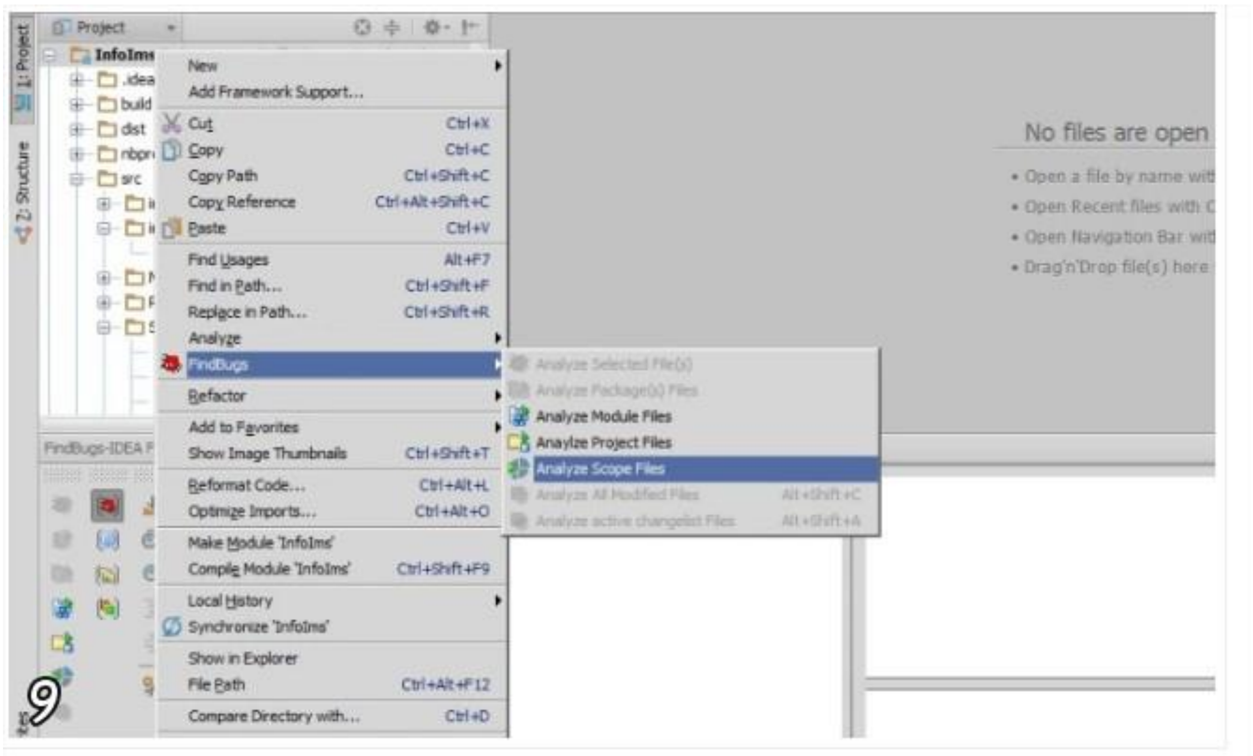
☑ **Find Security Bugs** (com.h3xstream.findsecbugs)
*file:/C:/Users/Malith%20Oshan/.IdeaIC12/config/plugins/FindBugs-IDEA-0.9.994/customPlugins/findsecbugs-plugin-1.2.1.jar*Find Security Bugs has rules that detect vulnerabilities, and also has rules that highlight parts of the source code that should be analyzed during a security audit.
Website: https://github.com/h3xstream/find-sec-bugs

Now we have successfully installed FindBugs plugin in IntelliJ IDEA and also have enabled the FindSecurityBugs plugin in it. Let's perform a static code analysis and get to know all the bugs we have in the code.

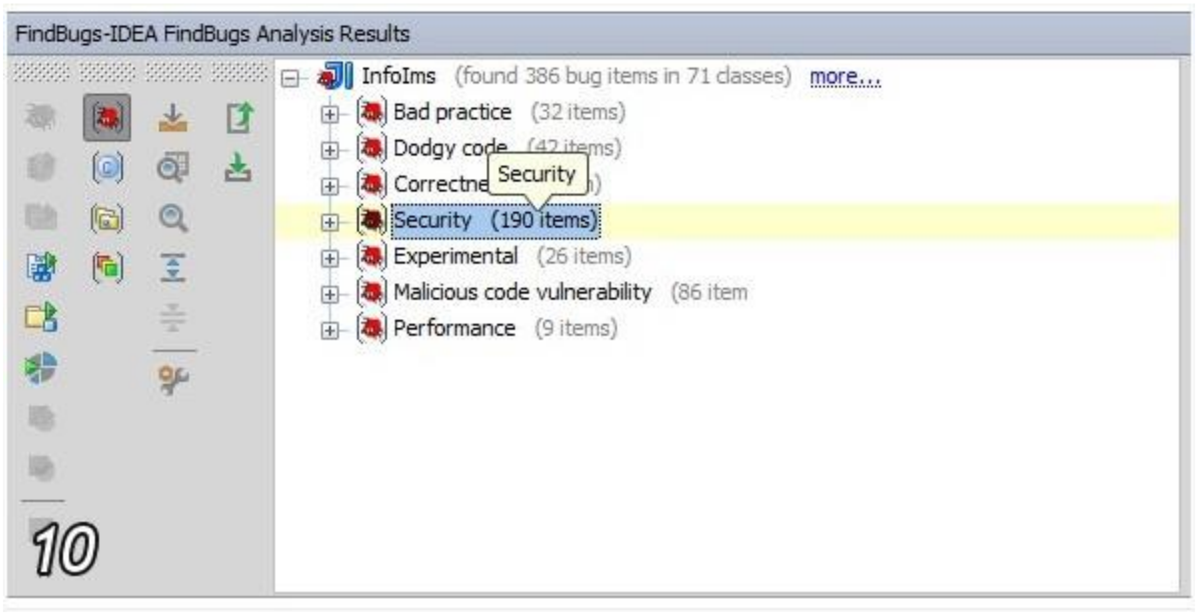
For this demonstration, I am using the OWASP Web Goat project which is a Java web application developed for security testing purposes which is plagued with lots and lots of security bugs.

I have opened the Web Goat project in IDEA and for running FindBugs on the project, I right click on the project and go to FindBugs -> Analyze Scope Files. With this, the scanning will happen only under the selected folder. You can also go with Analyze Module Files which would scan the particular module you have selected and also

Analyze Project Files which would scan the entire project.



Once the static scan is completed, you can see the identified bugs in FindBugs-IDEA panel. Since we have enabled the FindSecurityBugs plugin, it will list all the identified security issues under the Security category.

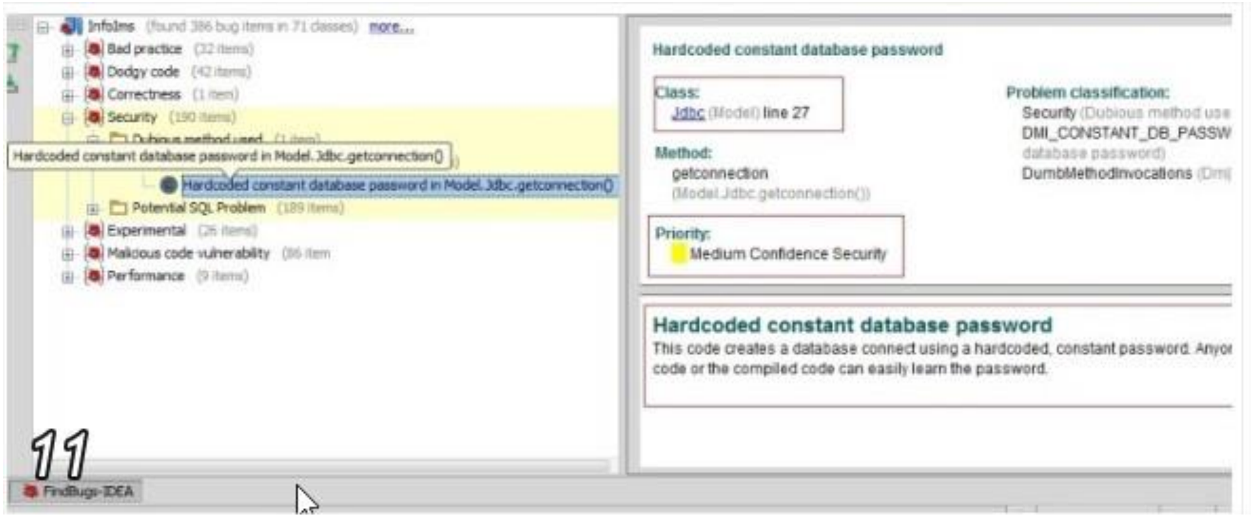


FindBugs-IDEA FindBugs Analysis Results

```
InfoIms (found 386 bug items in 71 classes) more...
  Bad practice (32 items)
  Dodgy code (42 items)
  Correctne Security )
  Security (190 items)
  Experimental (26 items)
  Malicious code vulnerability (86 item
  Performance (9 items)
```

According to my project that identified security bugs divided in to two main categories. First one is Dubious Method Used and the other one is Potential SQL Problem. These Categories differ from project wise. In this analysis it is describe only two main Security issues as below.

1) So in the first category contains one security bug it showed as following image.



```
InfoIms (found 386 bug items in 71 classes) more...
  Bad practice (32 items)
  Dodgy code (42 items)
  Correctness (1 item)
  Security (190 items)
    Dubious method used (1 item)
Hardcoded constant database password in Model.Jdbc.getconnection()
      Hardcoded constant database password in Model.Jdbc.getconnection()
    Potential SQL Problem (189 items)
  Experimental (26 items)
  Malicious code vulnerability (86 item
  Performance (9 items)
```

Hardcoded constant database password

Class:
Jdbc (Model) line 27

Method:
getconnection
(Model.Jdbc.getconnection())

Priority:
🟨 Medium Confidence Security

Problem classification:
Security (Dubious method use
DMI_CONSTANT_DB_PASSW
database password)
DumbMethodInvocations (Dm

Hardcoded constant database password

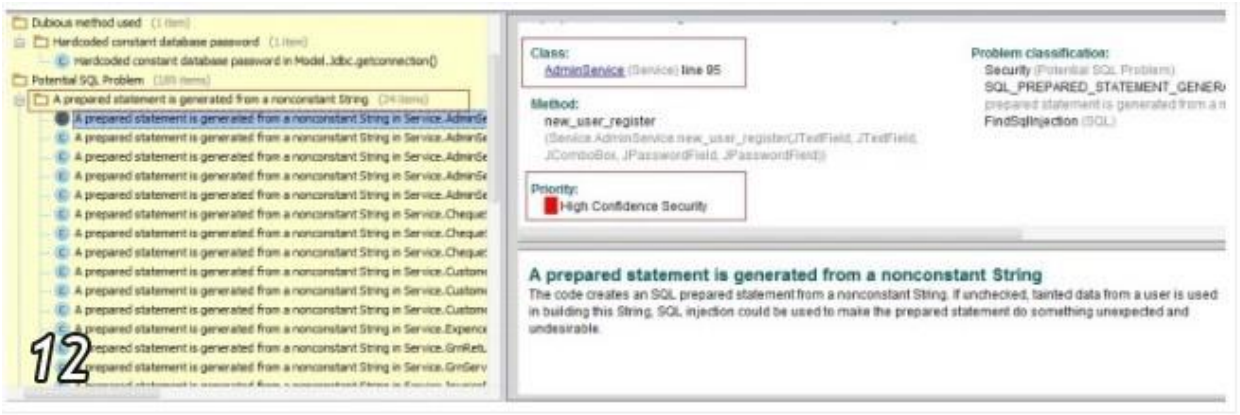
This code creates a database connect using a hardcoded, constant password. Anyor
code or the compiled code can easily learn the password.

- Solutions

There are many solution for above identified “Hardcoded Constant database Password”

1. The best one is use separate file with user name, password and other critical information related to the connection string and reads the file when the application starts. That is the only real way to prevent the password from leaking as a result of de-compilation.
2. Other one is org.jasypt.properties.EncryptableProperties class this for loading, managing and transparently decrypting encrypted values. http://www.jasypt.org/encrypting-configuration.html.

2) Now go through the second main category, in this category contains 189 bugs and its divide in to two categories again. First category “a prepared statement is generated from a non-constant string” contains 34 bug. In this every location has the same bug, but class and the code line is differ.



```
Dubious method used  (1 item)
  Hardcoded constant database password  (1 item)
    Hardcoded constant database password in Model.Jdbc.getconnection()
Potential SQL Problem  (189 items)
  A prepared statement is generated from a nonconstant String  (34 items)
    A prepared statement is generated from a nonconstant String in Service.AdminSe
    A prepared statement is generated from a nonconstant String in Service.AdminSe
    A prepared statement is generated from a nonconstant String in Service.AdminSe
    A prepared statement is generated from a nonconstant String in Service.AdminSe
    A prepared statement is generated from a nonconstant String in Service.AdminSe
    A prepared statement is generated from a nonconstant String in Service.Cheque
    A prepared statement is generated from a nonconstant String in Service.Cheque
    A prepared statement is generated from a nonconstant String in Service.Cheque
    A prepared statement is generated from a nonconstant String in Service.Custom
    A prepared statement is generated from a nonconstant String in Service.Custom
    A prepared statement is generated from a nonconstant String in Service.Custom
    A prepared statement is generated from a nonconstant String in Service.Expence
    prepared statement is generated from a nonconstant String in Service.GrnRet
    prepared statement is generated from a nonconstant String in Service.GrnServ
```

```
Class:
  AdminService (Service) line 95
Method:
  new_user_register
  (Service.AdminService.new_user_register(JTextField, JTextField,
  JComboBox, JPasswordField, JPasswordField))
Priority:
  🟥 High Confidence Security

Problem classification:
  Security (Potential SQL Problem)
  SQL_PREPARED_STATEMENT_GENER
  prepared statement is generated from a n
  FindSqlInjection (SQL)
```

A prepared statement is generated from a nonconstant String
The code creates an SQL prepared statement from a nonconstant String. If unchecked, tainted data from a user is used in building this String, SQL injection could be used to make the prepared statement do something unexpected and undesirable.

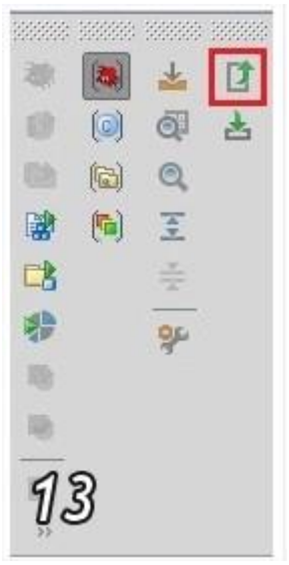
- Solutions

In this case we can’t ignore the warning because the priority is high. So the main issue for this code create SQL prepared statement from a Non-constant String.

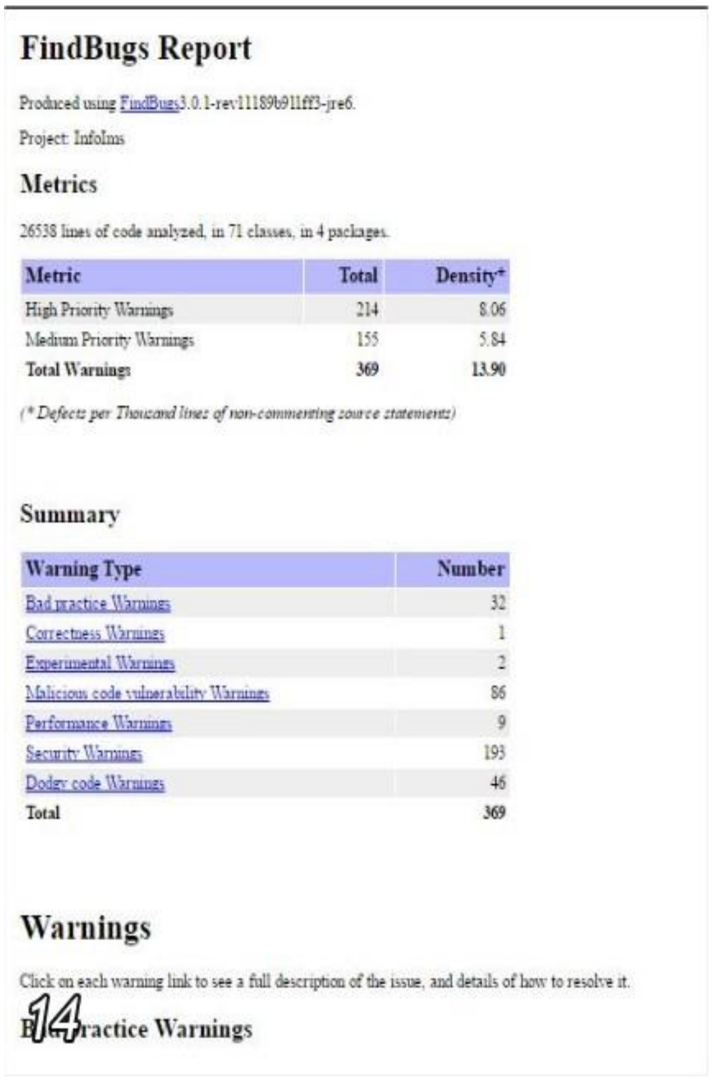
We need to declare the string as a

**Private static final

We can export the reported bugs for further analysis, for that click the Export Bug Collection to XML/HTML button.



A generated report would look like below.



FindBugs Report

Produced using FindBugs3.0.1-rev11189b911ff3-jre6.

Project: InfoIms

Metrics

26538 lines of code analyzed, in 71 classes, in 4 packages.

Metric	Total	Density*
High Priority Warnings	214	8.06
Medium Priority Warnings	155	5.84
Total Warnings	**369**	**13.90**

(Defects per Thousand lines of non-commenting source statements)*

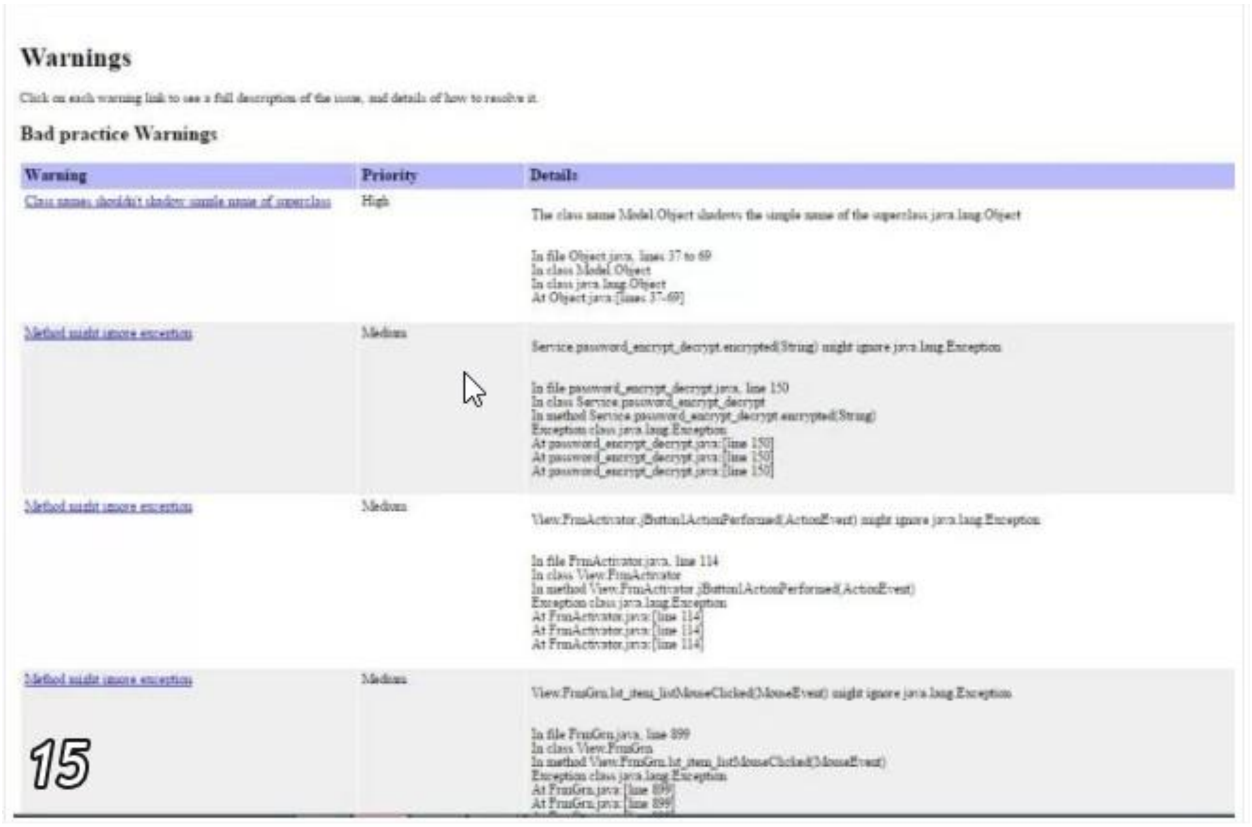
Summary

Warning Type	Number
Bad practice Warnings	32
Correctness Warnings	1
Experimental Warnings	2
Malicious code vulnerability Warnings	86
Performance Warnings	9
Security Warnings	193
Dodgy code Warnings	46
Total	**369**

Warnings

Click on each warning link to see a full description of the issue, and details of how to resolve it.

14 Bad practice Warnings



Warnings

Click on each warning link to see a full description of the issue, and details of how to resolve it.

Bad practice Warnings

Warning	Priority	Details
Class names shouldn't shadow simple name of superclass	High	The class name Model.Object shadows the simple name of the superclass java.lang.Object In file Object.java, lines 37 to 69 In class Model.Object In class java.lang.Object At Object.java:[lines 37-69]
Method might ignore exception	Medium	Service.password_encrypt_decrypt.encrypted(String) might ignore java.lang.Exception In file password_encrypt_decrypt.java, line 150 In class Service.password_encrypt_decrypt In method Service.password_encrypt_decrypt.encrypted(String) Exception class java.lang.Exception At password_encrypt_decrypt.java:[line 150] At password_encrypt_decrypt.java:[line 150] At password_encrypt_decrypt.java:[line 150]
Method might ignore exception	Medium	View.FrmActivator.jButton1ActionPerformed(ActionEvent) might ignore java.lang.Exception In file FrmActivator.java, line 114 In class View.FrmActivator In method View.FrmActivator.jButton1ActionPerformed(ActionEvent) Exception class java.lang.Exception At FrmActivator.java:[line 114] At FrmActivator.java:[line 114] At FrmActivator.java:[line 114]
Method might ignore exception	Medium	View.FrmGrn.lst_item_listMouseClicked(MouseEvent) might ignore java.lang.Exception In file FrmGrn.java, line 899 In class View.FrmGrn In method View.FrmGrn.lst_item_listMouseClicked(MouseEvent) Exception class java.lang.Exception At FrmGrn.java:[line 899] At FrmGrn.java:[line 899]

When writing code, it is important to use these kind of tools to identify potentials risks in your code and produce high quality code that is safe to use.

Reference

1. https://www.owasp.org/index.php/Static_Code_Analysis
2. http://findbugs.sourceforge.net
3. http://find-sec-bugs.github.io/

By S.H.M Lahiru Prabath Balasuriya.