Where Do Violations Come from?

You may be aware that there are very good open source tools available to analyze code and find anti-patterns. You may have even used them in the past. But when’s the last time you remembered to do it? Even if you are analyzing regularly on your own, what about your teammates? And are you all using the same tools? With the same options? Chances are the answer is no.

Sonar comes to the rescue. It provides a few rules of its own, but the majority of the rules it uses come from those major rules engines: Findbugs, PMD, and Checkstyle. In this article, we’ll take a brief look at rule profiles—the sets of rules your code is being measured against in an analysis—and how to change the defaults.

Picking a rule profile

Even though Sonar leverages Findbugs, PMD, and Checkstyle, it doesn’t turn on all three tools full-bore. Instead, it packages selected rules from each tool into what it calls profiles. Multiple profiles can exist per language and one profile is always set as a default for that language. New projects and projects that haven’t been specifically pinned to a rule profile, “unassigned” projects, are measured against their language’s default.

For Java, Sonar provides three profiles: Sonar Way, Sonar Way with Findbugs, and Sun checks. It sets Sonar Way as the default.

The Sun checks rule set is a small one, weighing in at only 58 rules. All it does is check source code against the Sun Java coding style conventions. For example, it checks member and class name capitalization, curly brace position and the use of spaces.

The difference between the other two—the two versions of Sonar Way—is the inclusion of Findbugs rules. And it’s a big difference. Checkstyle and PMD scan your uncompiled .java files. Findbugs runs against your compiled byte code, the .class files. During a Sonar analysis, if Findbugs is invoked (if any of its rules are included in the profile), Findbugs performs a static analysis of every possible path through the program—no stone unturned and no execution necessary.

Note Because Findbugs analyzes your compiled byte code, and not the Java source, the violations it finds may sometimes be shown in Sonar attached to the wrong line of code. When that’s the case, look one or two lines above and below the line Sonar flags, and you’ll find the real source of the problem, as shown in figure 1.

For source code, sample chapters, the Online Author Forum, and other resources, go to http://www.manning.com/papapetrou/
Sometime Findbugs flags the wrong line with the violation. It makes that occasional mistake because it’s working from the compiled byte code, not the .java files. When this happens, look a line or two above and below to find the real problem.

Findbugs’s bugs are the “best” ones, the ones you really want to catch because they find the issues that are the mostly likely to lead directly to bad program behavior. For instance, “Correctness—Null value is guaranteed to be dereferenced” comes out of Findbugs, and without Findbugs in your profile, you’re going to have a much harder time finding and fixing these rotten apples.

Clearly, you want to use the Findbugs rules if you have access to the .class files during analysis. Unfortunately, since that access can’t be assumed, Sonar’s out-of-the-box default is the less demanding Sonar Way profile.

Assuming that you’ve got the .class files handy, you’ll want to use the Sonar Way with Findbugs profile instead of the out-of-the-box default. The most expedient way to do that is to change Sonar’s default Java profile.

**Viewing profiles and changing the default**

Changing a language’s default profile is as easy as the click of a button for a logged-in administrator. The “Configuration” link at the top-right of the screen takes any user to the list of profiles, as shown in figure 2.

![Configuration link](image)

Figure 2 Any user can see the rule profiles by using the Configuration link at the top-right of the screen.

This means that any user can peruse the profile his code is being measured against (it’s listed in the Description widget on the dashboard.) But, only administrators see the two columns of controls shown on the right in figure 3.

For source code, sample chapters, the Online Author Forum, and other resources, go to [http://www.manning.com/papapetrou/](http://www.manning.com/papapetrou/)
For source code, sample chapters, the Online Author Forum, and other resources, go to http://www.manning.com/papapetrou/
customizing your rules. The default rule profiles Sonar provides can’t themselves be edited, but they can be copied and those clones can be freely edited by an administrator. For details, see chapter 12 of Sonar in Action, “Rule and Profile Administration,” for an in-depth discussion of profile management.

Summary
Sonar scans your program code for anti-patterns, and reports each instance as a rule violation. Not all violations are bugs, but every violation needs further attention.

Rules come from existing tools like Findbugs, PMD and Checkstyle as well as from Sonar itself. Rules are packaged into profiles, and for each language Sonar can analyze there will be at least one profile. Among the profiles for a language, there will always be one marked as default. An administrator can easily change the default profile.

If the default profile is not appropriate for your project, an administrator can easily assign the project to a different one.

For source code, sample chapters, the Online Author Forum, and other resources, go to http://www.manning.com/papapetrou/
Here are some other Manning titles you might be interested in:

- **Entity Framework 4 in Action**
  Stefano Mostarda, Marco De Sanctis, and Daniele Bochicchio

- **C# in Depth, Second Edition**
  Jon Skeet

- **Dependency Injection in .NET**
  Mark Seemann

Last updated: June 13, 2012

For source code, sample chapters, the Online Author Forum, and other resources, go to [http://www.manning.com/papapetrou/](http://www.manning.com/papapetrou/)