2018开源静态分析工具-第二部分-java

niexinming / 2018-10-24 15:10:00 / 浏览数 3148 安全工具 工具 顶(0) 踩(0)

翻译自：https://medium.com/@prasincs/open-source-static-analysis-for-security-in-2018-part-2-java-f26605cd3f7f
翻译：聂心明
昨天，我讨论了最好用的python开源静态分析工具。那java呢？尽管所有人都讨厌它，但这个语言依然处在TIOBE index（https://www.tiobe.com/tiobe-index/ ）的榜首。常言道，这通常是Bjarne Stroustrup说的（但是他否认）
世界上只有两种编程语言，一种是一直有人抱怨它不好，另一种是完全没有人用它。

Java虽然一直被人诟病，但是人们却用它写服务器端程序，或者安卓程序，或者其他的。我想JVM是最可靠的VM实现之一。
所以，今天要讲关于静态分析工具的什么故事呢？如果你想用TLDR版本的--可以使用spotbugs和pmd，我将进一步细致的讲解它是如何的工作的。但是我更愿意讲一些关于

依赖

我在上一篇文章（ https://medium.com/@prasincs/open-source-static-analysis-for-security-in-2018-part-1-python-348e9c1af1cd ）中谈论了安全的包，那么在java中怎么检测呢？可以用OWASP的依赖检测工具（ https://www.owasp.org/index.php/OWASP_Dependency_Check ），这个工具既可以独立运行，又可以作为maven 插件，甚至最近它被集成到了Jenkins中（https://plugins.jenkins.io/dependency-check-jenkins-plugin ）

```
<plugin>
   <groupId>org.owasp</groupId>
   <artifactId>dependency-check-maven</artifactId>
   <version>3.1.2</version>
   <executions>
       <execution>
           <goals>
               <goal>check</goal>
           </goals>
       </execution>
   </executions>
</plugin>
```

对于maven 你至少要添加上面的代码，现在你只要运行`mvn verify`,然后就能得到报告，或者运行`mvn dependency-check:check`,产生的报告默认保存在target/dependency-check-report.html。
从可实施性到可关注性，都是难以置信的好，你能完整的看到哪些包是存在安全问题的，并且放心，每一个脆弱点都有CVE编号，而且会把相关的链接放进去。

Scan Information (show all):
- dependency-check version: 3.1.2
- Report Generated On: Apr 19, 2018 at 08:50:16 -07:00
- Dependencies Scanned: 151 (121 unique)
- Vulnerable Dependencies: 7
- Vulnerabilities Found: 14
- Vulnerabilities Suppressed: 0
- ...

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	CPE	Coordinates	Highest Severity	CVE Count	CPE Confidence	Evidence Count
jackson-databind-2.7.6.jar	cpe:/a:fasterxml:jackson-databind:2.7.6 cpe:/a:fasterxml:jackson:2.7.6	com.fasterxml.jackson.core:jackson-databind:2.7.6 ✓	High	5	Highest	39
logback-core-1.1.7.jar	cpe:/a:logback:logback:1.1.7	ch.qos.logback:logback-core:1.1.7 ✓	High	1	Low	30
jetty-xml-9.3.9.v20160517.jar	cpe:/a:eclipse:jetty:9.3.9.v20160517 cpe:/a:jetty:jetty:9.3.9.v20160517	org.eclipse.jetty:jetty-xml:9.3.9.v20160517 ✓	Medium	1	Low	39
netty-3.6.1.Final.jar	cpe:/a:netty_project:netty:3.6.1	io.netty:netty:3.6.1.Final ✓	Medium	3	Highest	25
dropwizard-views-1.0.2.jar	cpe:/a:views_project:views:1.0.2	io.dropwizard:dropwizard-views:1.0.2 ✓	Medium	2	Low	20
logback-ext-core-1.0.3.jar	cpe:/a:logback:logback:1.0.3	ee.wrisk.logback:logback-ext-core:1.0.3 ✓	High	1	Low	23
protobuf-java-3.1.0.jar	cpe:/a:google:protobuf:3.1.0	com.google.protobuf:protobuf-java:3.1.0 ✓	Medium	1	Highest	29

这个插件会显示一个图表，目的是追踪每一个需要处理的依赖问题。如果你不想成为另一个 Equifax（https://arstechnica.com/information-technology/2017/09/massive-equifax-breach-caused-by-failure-to-patch-two-month-old-bug/ ），我建议你用这个。

IDEs

我拒绝使用Eclipse，但是我介绍的第一个静态工具实际上就是这个IDE的插件（它也可能装在Netbeans 上面），因为如果没有IDE，java几乎没有生产力。我发现IntelliJ's 内置的代码检查工具真的非常有用。对于IntelliJ 和JetBrains 等主要工具来说，有一些集成性是可用的，它们了解jvm并且知道如何在上面写一种类型的语言（ https://kotlinlang.org/ ）

测试覆盖工具

我发现，JaCoCo
能尽可能的覆盖测试大量你想要检查的代码，而且还会有一些基础性的检查。我提到它是因为它经常灵巧的发现一些代码路径，而这些路径如果没有被单元测试覆盖到的话，
http://www.baeldung.com/java-static-analysis-tools ）
在这一部分我会用到一些闭源工具，老实的说，他们似乎不值得花费。

FindBugs + FindSecBugs

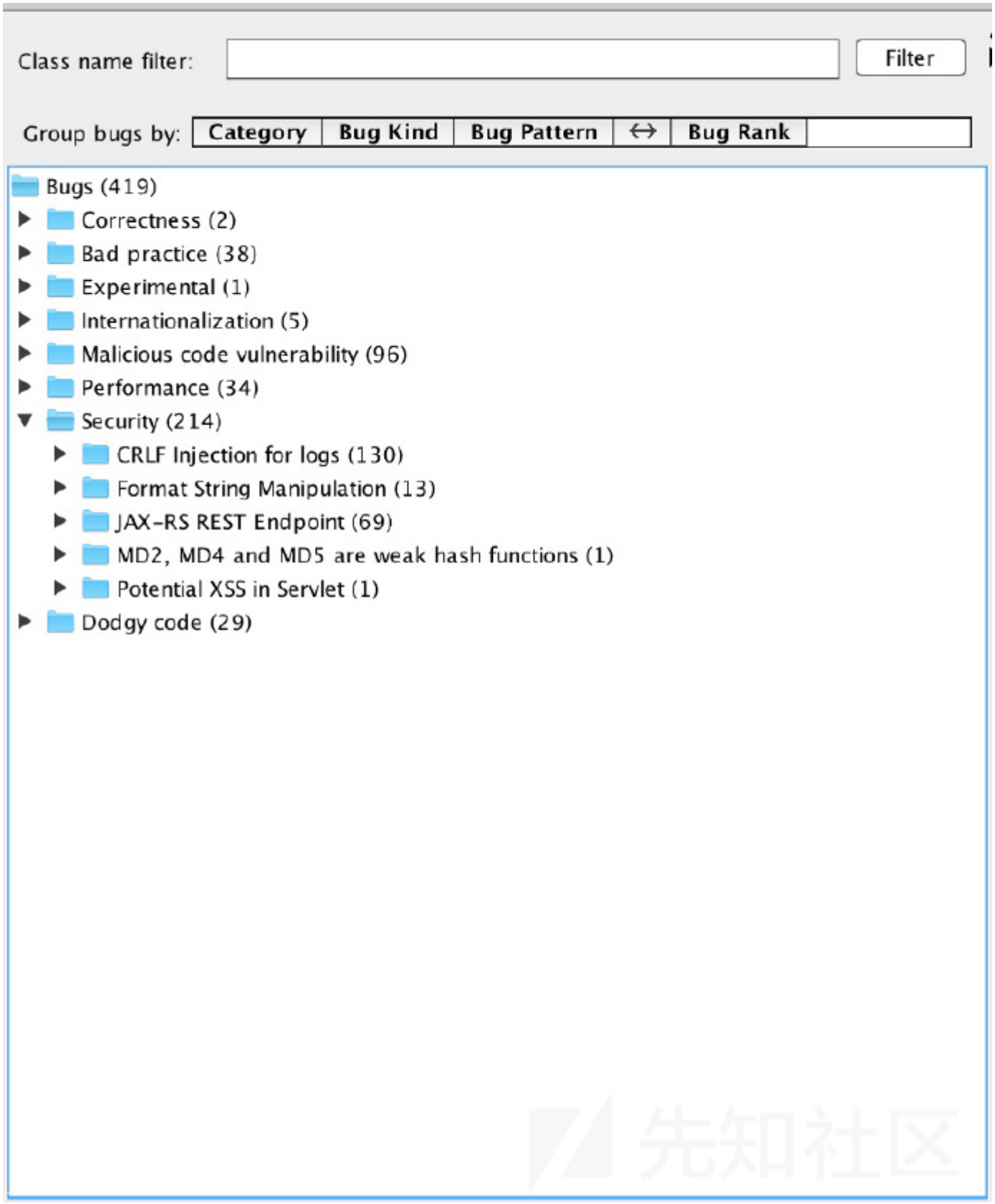
Findbugs
是第一个工具，在2016年几乎每一个人都建议我去使用，最后我放弃它了，是因为它不再被维护了，并且我厌倦了各种报错和bug。这似乎花费了我很多时间去讨论这些问题
http://findbugs.sourceforge.net/
在处理OWASP 漏洞的同时，我还需要解决许多的bug（况且，OWASP Top 10依然是最赚钱的产品），使用FindSecBugs（
https://github.com/find-sec-bugs/find-sec-bugs/ ），会稍微好一点。与现有工具集成会相对简单。SonarQube等工具可以很好地集成FindSecBugs。

SpotBugs + FindSecBugs

我发现可以将静态分析工具的可维护性和实用性做一个平衡。我通常用下面的方式将这个工具集成到maven，我已经有很长时间没有为Gradle
担心了。这里给你们一些有用的东西，把下面的代码放在<build><plugins>之间：

```
<!-- SpotBugs Static Analysis -->
            <plugin>
                <groupId>com.github.spotbugs</groupId>
                <artifactId>spotbugs-maven-plugin</artifactId>
                <version>3.1.1</version>
                <configuration>
                    <effort>Max</effort>
                    <threshold>Low</threshold>
                    <failOnError>true</failOnError>
                    <!--<includeFilterFile>${session.executionRootDirectory}/spotbugs-security-include.xml</includeFilterFile>
                    <excludeFilterFile>${session.executionRootDirectory}/spotbugs-security-exclude.xml</excludeFilterFile>-->
                    <xmlOutput>true</xmlOutput>
                    <!-- Optional directory to put spotbugs xdoc xml report -->
                    <xmlOutputDirectory>target/site</xmlOutputDirectory>
<plugins>
                        <plugin>
                            <groupId>com.h3xstream.findsecbugs</groupId>
                            <artifactId>findsecbugs-plugin</artifactId>
                            <version>LATEST</version> <!-- Auto-update to the latest stable -->
                        </plugin>
                    </plugins>
                </configuration>
            </plugin>
```

现在你运行mvn
spotbugs:check,然后它能有效地中断你的编译。你可以随意修改配置文件或者用pom强制编译或者在可信的环境中打包你的项目。比如，我可能允许使用快照生成一个小
你可以用mvn spotbugs:gui这个指令打开一个gui界面



现在虽然还是有一些误报，但是这是一个很好的开端。

PMD

我没有像使用SpotBugs那样使用PMD，这是一个成熟和稳定的静待代码检查平台。特别地是，它可以发现复制粘贴的代码（这些代码可能和与原来的代码具有相似的漏洞）
https://maven.apache.org/plugins/maven-pmd-plugin/
我没有在实际项目中用过PMD，所以没法提供更一步的信息，但是，如果我有的话，那么我会把示例代码放在这里。
虽然这些工具有了很大改善，但是还是有很大提高空间的，现在有一些开发者希望通过自己的力量帮助改善这些工具，我仍然希望从OWASP Top
10的开源工具区中听到一些好消息。我也已经熟悉了一些Java静态分析的闭源软件--像Coverity一样的Veracode和Synopsis工具，如果你愿意花一点小钱，SecureAssist也
最后，我发现单独的工具似乎不能应付所有的事情，因为：

1. 数据流很重要
2. 运行时态的改变很重要

3. 依赖的检查很重要
4. 锁很重要
5. 可变的所有权/漏洞很重要

如果你浏览 OWASP静态代码分析（ https://www.owasp.org/index.php/Static_Code_Analysis
）页面，你会发现很多静态分析的方法。我认为随着更好的工具和许多技术的出现，我们会越来越好，这些技术来自强类型的功能语言，如ML到主流语言。Java正在慢慢展

上一篇：jQuery-File-Uploa... 下一篇：PHP反序列化的一些例子

1. 0 条回复
   - 动动手指，沙发就是你的了！