恶意代码分析/检测的工具 -- 白帽子

有利用md5 检索的，有利用特征码检索的

1. **Codepecker 源代码缺陷审计 系统&平台（国内）**

**云平台介绍**

1. [Codepecker 国内第一款成熟的源代码缺陷审计云平台](http://www.cloudcp.cn/login.action)

2. Codepecker 产品简介

1）应用系统中在开发过程中，程序代码是否符合安全性要求，程序中是否存在安全漏洞，需要对应用系统程序原代码采用一定的技术手段和方法进行审核，预先发现其中的安全漏洞和具有潜在威胁的地方、提供相应的代码完善建议，有针对性的对用户的系统操作人员、开发人员和测试人员提供源代码安全培训服务

2）采用业界领先的源代码静态分析技术开发的一款针对源代码缺陷进行静态分析检测的产品，是国内第一款成熟的源码缺陷分析产品。

它能够高效的检测出软件源代码中的可能导致严重缺陷漏洞和系统运行异常的安全问题、程序缺陷，并准确定位告警，从而有效的帮助开发人员消除代码中的漏洞、减少不必要的软件补丁升级，为软件的信息安全保驾护航。

3. 目标客户：大中小型企业、政府机构、军队

4. 产品优势

1）CodePecker支持的语言种类多

2）能够全面的发现软件代码中的缺陷，包含软件安全漏洞、软件代码质量问题、还能狗发现编程中违反编程规则的情况

3）友好的图形分析界面

4）支持分析百万行级别的源代码

5）快速分析检测缺陷，低误报、漏报率

6）与多种主流IDE开发环境的集成

7）快速部署，不依赖于特定的操作系统

8）自动化的任务管理：无缝对接版本管理系统，完成自动/定时代码的扫描

1. 特点

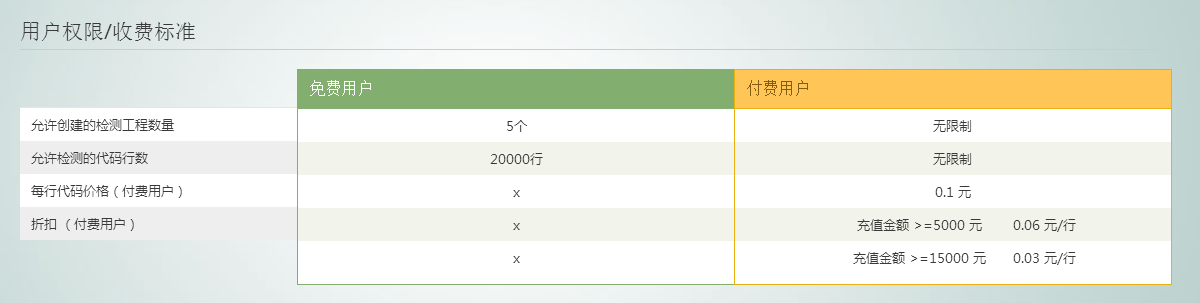
1）团队级代码扫描：隔离用户，支持多用户、多任务的并发缺陷分析

2）自动化的任务管理

3）报表分析：深度分析缺陷数据，形成缺陷数据分布图和趋势图，提供函数调用图以供参考

4）框架组件分析：支持如Struts、Spring、Hibernate等主流框架的漏洞分析

6. 收费标准



**单机版介绍**

1. 完全自主知识产权
2. 友好的用户界面

1）CodePecker采用全中文界面，全程中文界面支持，无论是操作界面、支持的文件目录类型，还是输出的报告均为中文，大大提升了国内用户的使用方便程度，从而大幅提升了工作效率。而国外同类产品都是英文的，只能输出英文的报告，有些中文目录也无法支持。

2）CodePecker在使用过程中，操作简单易用，不需要复杂的检测流程；检测结果简单明了，并有详细的缺陷分析，同时提供了缺陷分析的追踪定位，用户只需要简单的鼠标操作，就能够对缺陷传播定位。

3）鉴于开发人员对信息安全知识的了解，在缺陷类型中会有详细的缺陷点评，即使开发人员不熟悉此缺陷类型，通过缺陷点评，也能迅速的了解掌握此类缺陷。

1. 自定义的缺陷类型检测规则：

1）系统本身提供了25个大类，169种的缺陷类型检测，用户可以根据具体的需求对被检测代码做一个全量分析；

2）也可以根据业务需求，针对自己系统关心的缺陷进行定制检测。如在大型应用系统中，存在各系统中，存在各种级别的缺陷类型，检测结果可能偏多，会干扰错误排查，用户可只针对高危或者某几类类缺陷做有针对性的深度检测，只关注特定的缺陷类型，从而达到检测效果。-- 过滤器



1. 支持检测多种缺陷和质量类型：

1）CodePecker团队成员有着多年的源码安全检测经验，依靠业内知名专业的安全团队的研究，CodePecker漏洞知识库已包含多种语言多达数百条缺陷类型，每条漏洞都有详尽的描述和修补建议，同时积极与国际化接轨，大部分缺陷类型都可以映射到CVE和OWASP等权威国际安全组织公布的缺陷分类中。

2）缺陷类型涵盖了常见操作系统、数据库、Web工程和应用程序的绝大多数可以远程利用的漏洞以及本地安全漏洞。缺陷检测覆盖常见的多种语言、多种缺陷类别，包括跨站注入、Sql注入、拒绝服务等高危缺陷漏洞类型，也包括空指针引用，资源为释放、变量未初始化等代码质量缺陷类型。

3）从多个维度全面覆盖代安全问题，并积极更新最新研究结果和关注国内外最新安全研究动态，同步更新研究成果，保证缺陷知识库内容的覆盖广度和深度。

1. 高效快速的缺陷分析：

1）通过优化的数据流分析技术、缺陷类型的智能识别、检测规则依赖关系等源码扫描技术的运用；

2）加上安全团队根据多年的源码检测经验和国内外安全信息缺陷结果，以及完善的缺陷检测规则；

CodePecker缺陷检测软件在源码检测速度和检测结果的准确性做到了一个很好的平衡，既保证了检测的速度，又保证了缺陷检测的质量。

1. 低误报率和漏报率：

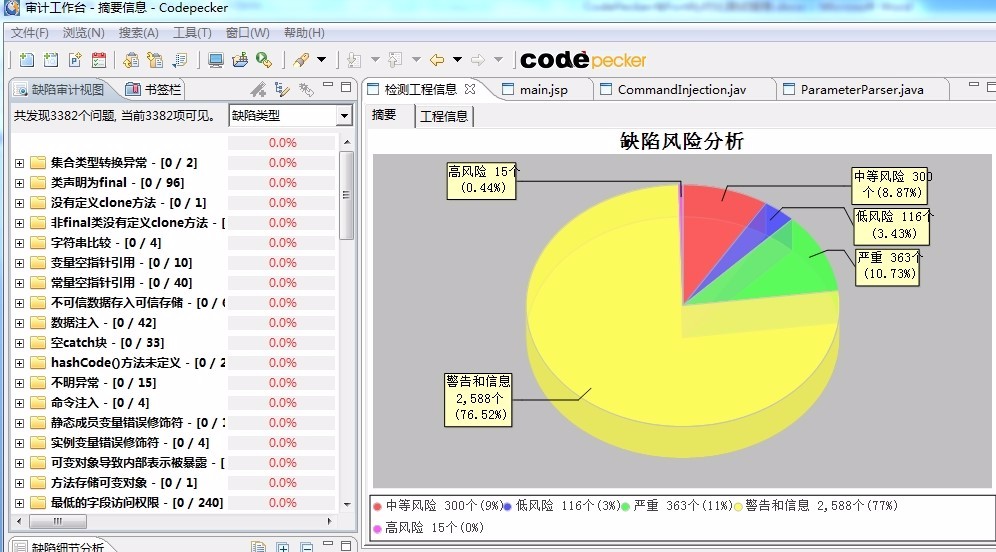
采用业内领先的深度缺陷扫描分析技术，CodePecker软件对同样的目标系统进行检测时，能提供过程内（Intra-procedure）、过程间（Inter-procedure）等各种层次的分析，全面深入地开展缺陷检测，全面降低了检测结果中的误报率和漏报率，检测精度、准度高。

1. CodePecker字节码，不需源代码也能检测：

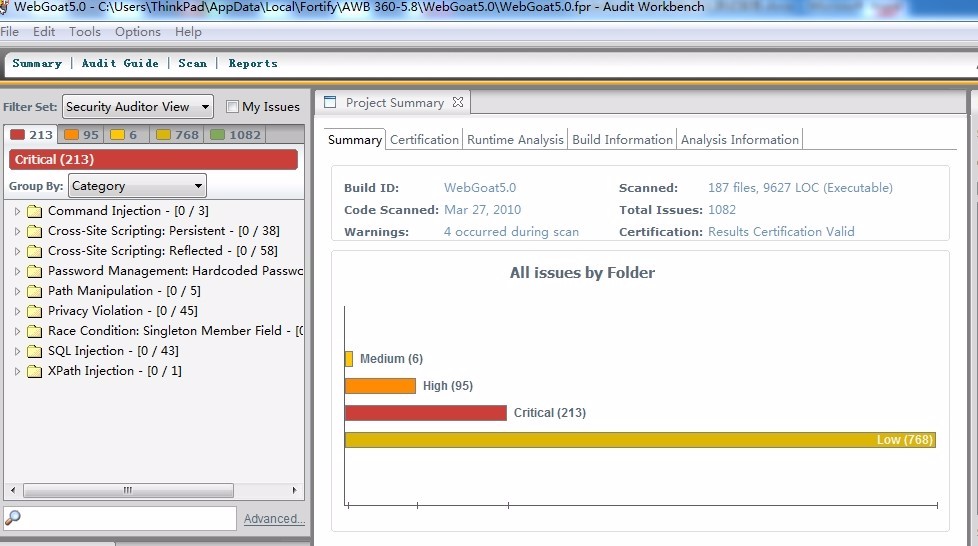
CodePecker软件源代码扫描系统将于2017年3月重大升级，这次升级主要是在源代码缺陷扫描基础上，增加了JAVA软件字节码缺陷分析工具。字节码分析工具不需要源代码，仅根据已发布的应用程序的字节码文件，就可以识别软件应用的安全问题。同时，由于可以不提供源代码，也为更大范围的代码覆盖提供了可能性。

**Codepecker源代码检测的比较 vs Fortify SCA**

1. 详细对比 <http://tieba.baidu.com/p/4873144998>
2. 整体对比 “Codepecker结果汇总图”



“Fortify检测结果汇总图”



**二、Coverity 代码静态安全检测（国外）**

**三、FindBugs 开源、静态分析工具（国外）**

**[官方主页](http://findbugs.sourceforge.net/)**

1. **FindBugs的优势**

1）“伪问题”的忽略和放弃（“伪问题”是部分代码质量工具容易为开发人员提供大量但并非真正问题的问题）；

2）它不注重样式或者格式，试图只寻找真正的缺陷或者潜在的性能问题；

1. **FindBugs是什么？**

1）静态分析工具，检查JAVA .class类或者JAR文件，将字节码与一组缺陷模式进行对比以发现可能的问题

2）Findbugs自带检测器，其中有60余种Bad practice，80余种Correctness，1种 Internationalization，12种Malicious code vulnerability，27种Multithreaded correctness，23种Performance，43种Dodgy；

3）我们可以自己配置检查规则(做哪些检查,不做哪些检查)，也可以自己来实现独有的校验规则(用户自定义特定的bug模式需要继承它的接口,编写自己的校验类,属于高级技巧)

1. **Java静态检查工具对比（直接copy，未印证）**

|  |  |  |
| --- | --- | --- |
| **工具** | **目的** | **检查项** |
| **FindBugs**  **检查.class** | 基于Bug Patterns概念，查找javabytecode（.class文件）中的潜在bug | 主要检查bytecode中的bug patterns，如NullPoint空指针检查、没有合理关闭资源、字符串相同判断错（==，而不是equals）等 |
| **PMD**  **检查源文件** | 检查Java源文件中的潜在问题 | 主要包括：  空try/catch/finally/switch语句块  未使用的局部变量、参数和private方法  空if/while语句  过于复杂的表达式，如不必要的if语句等  复杂类 |
| **CheckStyle**  **检查源文件**  **主要关注格式** | 检查Java源文件是否与代码规范相符 | 主要包括：  Javadoc注释  命名规范  多余没用的Imports  Size度量，如过长的方法  缺少必要的空格Whitespace  重复代码 |

1. **FindBugs的使用**

工具UI、Eclipse插件

**5. 详细说明 分析结果**

**[不同检测器 的列举实例](http://www.cnblogs.com/doit8791/archive/2012/10/22/2734730.html)（中）**

**[Bug类型及实例代码](http://www.cnblogs.com/hyddd/archive/2009/02/13/1390362.html)（英）**

Notes:其中Malicious Code指可能受到攻击的代码，与恶意代码检测不很契合！

**四、PMD开源、静态分析工具（国外）**

**[官网首页](http://pmd.sourceforge.net/pmd-5.0.0/)**

1. **PMD是一种开源、分析Java代码错误的工具**

1）与其他分析工具不同的是，PMD通过静态分析获知代码错误。也就是说，在不运行Java程序的情况下报告错误。

2）另外，它包含CPD（Copy-Paste-Detector），发现多种语言中的重复代码

1. **PMD附带了许多可以直接使用的规则**

利用这些规则可以找出Java源程序的许多问题，例如：

1）潜在的bug：空的try/catch/finally/switch语句

2）未使用的代码：未使用的局部变量、参数、私有方法等

3）可选的代码：String/StringBuffer的滥用

4）复杂的表达式：不必须的if语句、可以使用while循环完成的for循环

5）重复的代码：拷贝/粘贴代码意味着拷贝/粘贴bugs

6）循环体创建新对象：尽量不要再for或while循环体内实例化一个新对象

7）资源关闭：Connect，Result，Statement等使用之后确保关闭掉

1. **此外，用户还可以自己定义规则**

检查Java代码是否符合某些特定的编码规范。例如，你可以编写一个规则，要求PMD找出所有创建Thread和Socket对象的操作。

1. **工作原理**

PMD的核心是JavaCC解析器生成器。PMD结合运用JavaCC和EBNF（扩展巴科斯-诺尔范式，Extended Backus-Naur Formal）语法，再加上JJTree，把Java源代码解析成抽象语法树（AST，Abstract Syntax Tree）。

