静态源代码安全扫描工具测评基准 V2.0

文档描述	静态源代码安全扫描工具测评基准
最新版本	2.0
发布日期	2023 年 5 月 30 日
指导单位	OWASP 中国、ASA 应用安全联盟

目录

一、 背景介绍

（一）目的

静态源代码安全扫描工具或静态应用安全测试工具（SAST）是用于分析应用程序源代码、字节码或二进制码，以识别其中的安全漏洞的工具。它们不需要在程序运行时进行分析和检测，能够基于源代码帮助提升代码安全审计的效率。因此，它们能够在软件开发阶段和测试阶段尽早进行代码审计工作，以发现和解决安全漏洞。

目前市场上存在众多商业和开源的源码扫描产品或工具供选择，企业对源代码安全的重视程度也在增加。然而，目前缺乏全面的静态源代码安全扫描工具测评基准，这使得企业在选择和采购产品时缺乏评判依据。

因此，本基准的目的是创建一个全面、客观、中立的源代码安全扫描工具评价基准，供企业、机构或团队在选择和采购静态源代码安全扫描工具时作为参考依据。

（二）范围

本基准的测评范围仅限于以静态方式（即不运行程序，针对源代码、二进制文件或编译后文件）检测源代码中安全缺陷或漏洞的软件工具。不包括扫描其他介质（如数据库、文档）或以动态方式执行代码的工具。

（三）技术背景

由于安全风险和漏洞的发现和修复在软件开发生命周期的早期介入可以获得较高的回报，越来越多的人开始关注安全左移（即安全工作在软件开发的早期

阶段开始介入）。然而，目前尚无法完全分析软件中存在的架构级别的安全、质量和正确性问题的安全分析和检测工具。而无论是人工安全测试还是动态漏洞测试所发现的漏洞都需要通过修改源代码来进行修复。通过静态分析技术分析源代码，安全漏洞和缺陷能够以更全面、更直接、更简单的方式呈现给安全人员和开发人员。

因此，静态源代码安全扫描工具在安全测试中始终占据重要位置。然而，要有效地使用此类工具，必须将源代码检测与软件设计相结合，通过新增、修改检测规则来提高源代码安全扫描的准确度与覆盖率。

二、 规范性引用文件

本基准的制定过程中参考的相关文件如下：

- 《GB/T 25069-2022 信息安全技术 术语》：关于信息安全技术术语的国家标准。
- 《GB/T 39412-2020 信息安全技术 代码安全审计规范》：关于代码安全审计规范的国家标准，提供了代码审计的指导原则和方法。
- 《OWASP Top 10：2021》：当前最常见的十大网络应用安全风险的指南，由全球开放式 Web 应用安全项目（OWASP）提供。
- 《OWASP ASVS》：OWASP 基金会制定的 Web 应用安全验证标准。它定义了不同级别(Level 1 到 Level 3)的安全要求，帮助企业和开发者构建更安全的 Web 应用。

除了上述文件外，本基准制定过程中还参考了以下内容：

- 行业标准和最佳实践：参考与静态源代码安全扫描相关的行业标准和最

佳实践。

- 开源项目和社区：参考开源静态源代码安全扫描工具的文档、指南和最佳实践。

三、 术语

安全（security）

对某一系统，据以获得保密性、完整性、可用性、可核查性、真实性以及可靠性的性质。

安全功能（security function）

为实现安全要素的要求，并正确实施相应安全策略所提供的功能。

风险（risk）

对目标的不确定性影响。

漏洞（vulnerability）

在信息系统中存在的弱点、缺陷或不合规定的组件，可以被恶意用户或攻击者利用，导致系统遭受未经授权的访问、信息泄露、数据损坏、服务中断或其他安全问题。

SAST（Static Application Security Testing）

静态应用程序安全测试技术

OWASP（Open Web Application Security Project）

开放式 Web 应用程序安全项目

四、 测评基准概要

测评基准是源代码安全扫描工具所应支持功能的类型，其功能的类型和能力决定了企业、机构或团队在进行源码安全扫描时候的适用程度、检测能力和匹配程度。

静态源代码安全扫描工具的测评基准包括：

- 部署环境
- 安全扫描
- 漏洞检测
- 源码支持
- 扩展集成
- 产品交互
- 报告输出

五、 测评基准说明

（一）部署环境

工具所能够支持的部署环境决定了工具在使用中对于不同使用环境和场景的适应能力和匹配能力，包括终端环境下个人用户的测试、使用，以及开发和测试环境下服务端环境的测试、使用。

1. 操作系统支持

工具应支持主流操作系统，以满足不同开发环境和测试环境的需求，包括但

不限于：

- Windows
- Mac OS
- Linux

2. 容器化支持

工具应提供容器化支持，以方便集成到开发和部署流程中，如 Docker。

（二）安全扫描

1. 扫描速度

扫描速度指的是针对源代码的漏洞扫描效率，在进行源代码包的安全漏洞扫描时，更快的扫描速度可以降低安全人员等待的时间，提高漏洞发现和反馈的效率。因此，源码安全扫描工具的扫描速度越快越好。

源代码扫描需要支持至少 3 种主流开发语言和不同代码量级别（千行、万行、百万行）的扫描。

单次代码扫描的时长会直接影响到开发和测试效率，不利于快速迭代和反馈，根据业内报告显示，单次代码扫描时长合理的速度如下：

项目规模	代码规模	单次扫描时长
小型项目	<50 万行	<2 小时
中型项目	50-500 万行	<8 小时
大型项目	>500 万行	<72 小时

2. 扫描配置

为了更好的适应和匹配不同的测试场景和扫描场景，提升工具的易用性和适应性。

静态源代码安全扫描工具的扫描配置应包含如下能力：

序号	能力	描述
1	定时扫描	能够对于同一扫描项目或源代码设定周期进行自动扫描，以定期扫描、发现同一源代码包变更后的安全漏洞识别。
2	并发扫描	对于项目较多的组织或企业并发扫描的能力非常重要，逐个排队扫描会影响漏洞的检测和发现周期，并发扫描能够最大化利用硬件的资源，提升漏洞检出效率。
3	增量扫描	增量扫描通常应用于迭代项目，通过扫描项目的变更部分代码降低漏洞扫描周期，减少不必要的代码扫描范围。

3. 漏洞误报率

漏洞误报率是指扫描结果中误报的漏洞占到所有检出漏洞数量的比率：

(误报的漏洞数量 / 所有扫描结果中的漏洞总数) * 100%

漏洞误报率是静态源代码安全扫描工具漏洞检测准确率的重要指标之一，也是用户使用 SAST 产品的主要顾虑。大量的漏洞误报会增加安全人员或开发人员的漏洞确认或排查时间和精力，从而增加项目安全漏洞修复的周期和成本，以及

降低相关人员对漏洞修复的积极性，增加安全漏洞修复的隐性成本，过高的误报率会影响开发人员对于静态源代码扫描工具的信心。

影响漏洞误报率的因素包括代码复杂度、代码框架，因此无法完全避免漏洞误报或保持非常低的误报率。基于行业相关研究，静态源代码安全扫描工具的漏洞误报率在 10%-20%之间或更低能够更好被开发人员接受。

4. 漏洞漏报率

漏洞漏报率是指扫描结果未包含的真实漏洞占到所有漏洞数量的比率：

(未发现的漏洞数量 / 所有真实的漏洞总数) * 100%

漏洞漏报率是静态源代码安全扫描工具漏洞检测准确性和可靠性的重要指标之一。如果漏洞漏报率过高，会遗漏未检出的漏洞，从而给应用程序的安全性带来风险。漏洞漏报率低，意味着漏洞检出结果遗漏的真实漏洞越少。因此，漏洞误报率和漏洞漏报率是安全检测工具或产品使用中用户最关注的两个指标。

根据行业内的调查报告，静态源代码安全扫描工具的漏洞漏报率在 20%-30%之间或更低能够被企业用户接受。

5. 编译代码支持

针对源代码的安全漏洞扫描会面临无源代码的情况，尤其是 C/C++的编译文件，或者是源代码项目中所包含的其他编译后代码的结果，如链接库、组件等。

因此，静态源代码安全扫描工具支持对代码编译后的二进制或字节码扫描，能够发现更多形态的静态文件中安全漏洞或风险，提高静态源代码文件的扫描覆盖率。

6. 移动应用支持

静态源代码安全扫描工具应能够支持对移动应用源代码的安全扫描，包括但不限于 iOS、Android 应用包。

7. 漏洞规则支持

在静态源代码安全扫描工具使用中需要根据不同的使用场景以及漏洞类型进行漏洞规则的自定义，以帮助使用者能够更高效地发现、处置漏洞，避免不必要的漏洞误报产生，或者处置超出能力范围的大量漏洞。

漏洞规则支持指的是工具应当具备漏洞规则自定义能力，包括：

（1）修改漏洞规则能力

当源码设计和开发层面已经使用了特定了方法和框架来缓解漏洞或规避漏洞，而工具没有能够正确识别对应类型的漏洞，则应当对漏洞检测规则进行修改以降低漏洞误报率。

下列是常见的源代码采取了特定设计和方法，需要做漏洞规则修改的场景：

- 特定的输入方法不被识别；
- 特定的输出方法不被识别；
- 程序特有的过滤函数不被识别；
- 程序特有的数据库操作 API 不被识别；
- 程序采用了工具不支持的框架；
- 规则逻辑错误；

（2）新增漏洞规则能力

在一些检测的场景中，工具无法利用已有检测规则发现漏洞，因此工具需要能够具备漏洞规则新增能力，增加对于新类型漏洞的检测、发现能力，定义的漏洞规则应当包括：

- 漏洞类型名称
- 漏洞类型描述
- 漏洞检测规则

如果是工具是以自定义编码语言进行漏洞规则定义，则应当提供详细的语言语法说明文档。

8. 漏洞标记能力

漏洞标记是指在静态源代码安全扫描工具能够自动对检出的漏洞进行标记、分类、归档功能。

（1）标记漏洞：将检测到的漏洞标记为已发现的漏洞，并提供漏洞的详细信息，包括漏洞的严重性、漏洞所在的代码位置、漏洞的类型等。

（2）分类漏洞：将漏洞按照等级、类型、归属等进行分类，以便使用者能够更好地了解漏洞的性质和影响。

（3）归档漏洞：将已修复的漏洞归档，以便使用者跟踪漏洞修复历史和分析漏洞趋势。

漏洞标记功能可以帮助使用者更好地管理和跟踪漏洞，提高漏洞管理的效率和准确性，同时帮助管理和分析漏洞相关的数据和信息。

（三）漏洞检测

1. 漏洞类型支持

代码扫描工具应涵盖系统设计、开发过现中常见的安全漏洞类型。安全漏洞有很多种分类方式，OWASP TOP10 是业界被普遍认可的基于行业反馈和趋势的漏洞分类方式。代码扫描工具应覆盖 OWASP Web TOP10 所包含的危害性最大或者攻击成本最低的漏洞：

（1） Broken Access Control（失效的访问控制）；

（2）Cryptographic Failures（加密机制失效）；

（3）Injection（注入）；

（4）Insecure Design（不安全的设计）；

（5）Security Misconfiguration（安全配置错误）；

（6）Vulnerable and Outdated Components（自带缺陷和过时的组件）；

（7）Identification and Authentication Failures（身份识别和身份验证错误）；

（8） Software and Data Integrity Failures（软件和数据完整性故障）；

（9）Security Logging and Monitoring Failures（安全日志记录和监控失败）；

（10）Server-Side Request Forgery（服务器端请求伪造）；

注：上述 10 类漏洞来源于 OWASP Web Top 10 2021 项目，该项目的安全问题会保持更新。

2. 漏洞信息支持

静态源代码安全扫描工具应对每一类安全漏洞提供清晰、易懂的漏洞描述以及对应的漏洞修复建议，便于使用者理解和修复安全漏洞。

漏洞信息应当包括：

- 漏洞名称
- 漏洞描述
- 漏洞影响
- 修复建议

3. 开发框架支持

静态源代码安全扫描工具应当支持不同类型开发框架的安全漏洞扫描，能够理解和解析各种语法结构和代码依赖关系，能够识别开发框架的相关信息，包括框架名称、类型、版本。

（四）源码支持

1. 开发语言类型

鉴于编程领域语言不断丰富，软件的编程语言组成日趋多样化，为了满足使用者对各类开发语言的扫描需求，静态源代码安全扫描工具应能够支持的开发语言种类应包含业界主流的开发语言。

目前业界常用的开发语言包括：

- Python

- PHP
- Java/JSP
- C/C++
- C#
- JavaScript
- .Net
- Objective-C
- Go
- Swift

2. 源码导入方式

源代码检测过程中可能会受到源码文件大小、源码来源限制、源码包文件格式等多种因素影响，静态源代码安全扫描工具的源码导入方式应当支持：

- 本地上传源码包（包括 RAR、Zip 等压缩文件格式）
- 代码仓库（包括但不限于 GIT、SVN 等）拉取
- 源码片段上传

（五）扩展集成

静态源代码安全扫描工具在开发流程或测试流程的使用中，具备扩展集成能力能够降低工具在开发、测试使用中的使用成本，提升开发、测试过程中的安全扫描效率，有利于 SDL 落地或 DevSecOps 的构建。

1. 源代码管理系统集成

静态源代码安全扫描工具应支持与源代码管理系统（或源代码托管平台）的集成。通过与版本管理工具如 Git、SVN 集成，在无需人为操作的情况下，由工具按照预先配置的版本管理方式完成源代码的获取。

源代码自动拉取是节省整个源代码检测周期人力、时间，并提升测试覆盖率的重要手段。与源代码管理系统集成的方式获取源代码文件也更符合安全要求，避免源代码手工上传过程中存在的代码泄露风险。

2. 缺陷跟踪系统集成

为了在漏洞扫描和分析结束后同步漏洞信息至开发人员进行漏洞修复，同时便于漏洞修复任务指派、跟踪漏洞修复情况，静态源代码安全扫描工具应提供与缺陷跟踪系统集成的能力，包括：

（1）单漏洞同步：将单个漏洞信息（至少包含漏洞名称、漏洞类型、漏洞描述、漏洞影响、漏洞位置、修复建议）提交、同步至缺陷跟踪系统；

（2）批量漏洞同步：批量把项目的漏洞信息（同上）提交、同步至缺陷管理系统；

3. 持续集成系统集成

在测试流程中，对于集成工具或集成系统的集成支持能够提升测试环节的源代码安全扫描效率，因此静态源代码安全扫描工具应持续集成系统(如 Jenkins)集成或对接能力。

（六）产品交互

为了适应不同的软件开发流程与工具使用环境，静态源代码安全扫描工具需要支持不同的交互模式。

交互模式	描述
图形界面模式	使用者通过客户端界面或浏览器界面完成工具的功能使用。 图形界面模式直观、易用、学习成本低。
命令行模式	使用者通过命令行方式完成工具的功能使用。 命令行模式需要使用者熟悉系统常见命令，并熟练命令行下的系统使用，学习成本较高。
IDE 插件模式	使用者通过安装工具的 IDE 环境插件，在集成开发环境中使用插件进行源代码安全扫描。

（七）报告输出

静态源代码安全扫描工具应该具有安全漏洞扫描结果的报告输出能力，输出的报告应当具至少包含漏洞统计（包含漏洞等级、漏洞类型）、漏洞位置、漏洞描述、代码片段、修复建议，帮助报告使用者了解源码漏洞的整体状况，以及快速定位漏洞并进行漏洞修复。

工具应当支持导出多种文件格式的报告，包括但不限于 PDF、HTML、Word、Excel 及 XML。

鸣谢

特向参与本项目的编写及审核人员致以诚挚的谢意：

谢立斐、做个好人、秦刚峰（华润数科）、土豆（猎豹移动）、米兰（好未来集团）、cumirror（腾讯云武汉）、无所不能的魂大人（雪球基金）、崖（亚马逊云科技）