ASC应用安全联盟安全基准项目

静态源代码安全分析工具测评基准

第一版

ASC应用安全联盟

ASC 静态源代码安全分析工具测评基准

v1.0

文档描述	静态源代码安全分析工具测评基准
版本	V 1.0
日期	2016 年 5 月 30 日
作者	徐瑞祝、谢春刚、靳超、王明、张龑

文档变更记录

序号	更改章节	更改内容	更改人	更改日期	备注

目 录

第一章 介绍.....5
1 目的.....5
2 范围.....5
3 技术背景.....5
4 术语.....6
第二章 技术能力标准.....7
1 部署环境.....7
1.1 部署模式.....7
1.2 工具架构.....8
1.3 工具安装支持.....8
1.4 运行时依赖.....9
2 License 认证方式.....9
3 技术能力标准.....10
3.1 支持的编程语言.....11
3.2 支持扫描编译后的代码.....12
3.3 支持的扫描环境.....12
3.4 支持的安全漏洞类型.....12
3.5 支持快速定位问题的功能.....13
3.6 支持分析结果的对比分析.....14
3.7 支持软件安全等级的评定和安全趋势的分析.....14
3.8 支持提供漏洞修复指导.....14
4 用户体验.....15
5 使用模式.....17
6 扫描能力.....18
6.1 基础扫描能力.....18
6.2 扫描速度.....22
6.3 扫描配置能力.....23
6.4 支持的分析引擎.....23
6.5 漏报率.....24
6.6 误报率.....24
7 报表能力.....25
7.1 支持基于角色的报表.....25
7.2 报表格式.....26
8 扩展能力.....26
8.1 与第三方工具集成能力.....26
8.2 与源码管理系统集成能力.....26
8.3 与缺陷跟踪系统集成能力.....26
8.4 与持续集成系统集成能力.....27
8.5 规则自定义能力.....27
8.5.1 修改漏洞规则能力.....27

8.5.2 新增漏洞规则能力 28
8.6 与黑盒安全检测工具的配合能力 28
8.7 客户化能力 28
8.7.1 界面客户化 29
8.7.2 功能客户化 29
8.7.3 报表自定义 29
第三章 附录 31
附录一：静态源代码安全分析工具非技术能力基准 31
1 静态源代码安全分析工具采购评估依据 31
2 软件供应商公司实力评估依据 31
3 价格评估依据 32
4 售前支持服务 33
5 售后支持服务 33
6 产品更新能力 34
附录二：引用 35
附录三：ASC 应用安全联盟介绍 36
1 联盟简介 36
2 联盟章程 36
3 联盟项目 37
4 联系我们 37

第一章 介绍

1 目的

静态源代码安全分析工具（SAST）是设计为分析应用程序源代码、字节码或二进制码以指出安全漏洞的技术。静态源代码安全扫描解决方案不需要在程序运行时分析应用程序，能够替代人工代码审查的方式，从而把代码审查推到软件开发生命周期更早的阶段。因此，静态技术可以极大的节省人力，为企业用户所青睐。

虽然目前市场上已有众多的商业与开源工具可供选择，但是由于缺乏全面的评估标准，专业透彻的评估很难实现，从而导致用户往往无法完全了解其中的优劣。

本书的目的是为了创建一个第三方的、中立的源代码安全分析工具检测基准，以供安全组织与专家对静态源代码安全审核工具进行全面及公正的评估。

2 范围

本书测评范围限于以静态的方式检测源代码中安全缺陷和潜在漏洞的软件工具。任何扫描别的介质（如数据库、文档），以及以动态的方式执行代码的工具均不在本书涵盖范围之内。

3 技术背景

在软件的开发生命周期中，对安全的关注重点越往前移，其投入回报比越高。但当前还没有任何分析和检测工具可以完全分析软件中存在的架构级别的安全、质量与正确性。这些属性应该在软件设计时考虑，但静态工具能够在一定程度上提供帮助。虽然人工测试或使用动态工具测试可以查看程序在运行时的安全状态，但是只有静态工具才能够检测恶意的后门。

不管是人工安全测试还是动态渗透测试发现的漏洞都需要通过修改代码来修复。通过静态分析技术分析源代码的安全缺陷能够以更直接的方式告诉开发人员需要修复的问题点。

静态源代码安全分析工具在安全评估中占有一席之地。但若要有效的使用该类工具，必须要与软件设计内容相结合，通过调整、新增规则以增加其精确度与覆盖率，否则其结果仍然需要大量人工去验证。

4 术语

表 1.1 术语表

英文名	中文名	解释
Static Code Analysis Tool	静态源代码安全分析工具	使用静态技术分析代码中存在安全缺陷的工具。
Static application security testing (SAST)	静态应用程序安全测试技术	在程序非运行时通过分析代码进行安全测试的技术。
Vulnerability	漏洞	可能被利用的缺陷。
False Negative	漏报	工具未指出一个确定存在有漏洞。
False Positive	误报	工具指出一个不存在的漏洞。
Weakness	缺陷	包含漏洞的一段代码。
POC	观点论证	为产品实施方案提供论证。
Scan	扫描	基于漏洞规则对源代码进行安全分析、发现安全问题，是静态源代码安全分析工具的核心工作形式。

第二章 技术能力标准

1 部署环境

部署环境是软件的物理载体，对部署环境有清晰的了解和精确良好的设计，能够帮助用户避免不必要的硬件采购成本，保证软件正常使用。

可以从以下几个方面了解部署环境：

- 部署模式
- 工具的架构
- 运行时依赖
- 工具安装支持

1.1 部署模式

厂商如果提供多种部署选项，应该说明每一种部署选项的不同，以便用户能够选择最适合自身的方案。

静态源代码安全分析工具的部署模式常见包括：

- **桌面(C/S)**：厂商发布一个安装包给用户，用户授权可以安装到一台或多台机器上。
- **私有云（B/S)**：部署在企业内部的服务器上，用户使用时，通过浏览器提交代码及查看扫描结果。
- **公有云（SaaS)**：部署在厂商服务器上，客户通过提交扫描代码在线获取扫描结果。

1.2 工具架构

评估工具架构主要是查看工具对硬件资源的利用能力、多任务处理能力等，这些能力将决定扫描能力的扩展性。常见的静态源代码安全分析工具可提供如下的能力：

硬件资源的利用能力：

- 利用多核 CPU
- 利用大内存

多任务处理能力：

- 实现并发扫描
- 多引擎部署

1.3 工具安装支持

工具应当提供安装使用辅助文档以指导产品的安装。安装指导应包括：

- 软硬件环境最小需求
- 详细的工具安装指引（包括插件等子系统）

如需额外的配置（如与子系统的配置）也应详细说明。

如果是基于 SaaS 的服务，应当提供详细的说明：

- 如何去启动服务
- 客户端浏览器支持列表
- 安全传输的保障手段

1.4 运行时依赖

厂商应当说明静态源代码安全分析工具或服务的运行时依赖：

- **依赖于编译：**要求代码互相之间的依赖关系必须完整，部分工具可能需要提供编译环境（如编译器或开发工具），或者是编译后的字节码。
- **不依赖于编译：**不要求扫描的代码必须编译通过，可以实现快速扫描。

2 License 认证方式

License 认证方式是使用环境管理（如硬件变更）的重要考虑因素。不同静态源代码安全分析工具的认证方式各异。

常见静态源代码安全分析工具 License 的认证方式包括：

表 2.1 License 认证方式

认证方式	描述	特点
服务器端连接认证	通过连接厂商的 License 服务器进行认证。	客户安全环境必须可以连接到互联网。
软硬件 License 码认证	基于本地硬件和操作系统环境，包括 CPU 型号、硬盘序列号、MAC 地址、操作系统类型等。	更换软硬件或硬件损坏时必须要更换 License。
用户名密码认证	基于操作系统登录名和机器名进行认证，基于这种认证方式的 License 一般具有时效性，过期 License 将会失效。	License 使用不可控，客户可以重复安装。
合同约束	以合同授权的形式给予用户使用权。	License 使用不可控。
硬件密钥认证	把 License 密钥存放于硬件中。	对 License 的控制较好，更换硬件时必须更换 License。

3 技术能力标准

技术能力标准是指静态源代码安全分析工具能所支持功能的类型。静态源代码安全分析工具的技术能力可决定对企业的适用性。例如在 Linux 下进行 C 和 C++开发的团队不能选择仅支持 Windows 环境或者仅支持 Java 源代码安全分析的工具。因此，技术能力标准是选择静态安全扫描工具必须考虑的硬性指标。技术能力标准主要从以下几个角度考虑：

- 支持的编程语言
- 支持的扫描环境
- 支持的缺陷类型

- 支持快速定位问题
- 支持分析结果的对比分析
- 支持软件安全等级的评定和安全趋势分析

3.1 支持的编程语言

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，静态源代码安全分析工具能支持的语言种类应包含业界主流的开发语言及其所组成完整软件所需的必要附属语言。

常用编程语言包括但不限于：

	Android	.Net	Object-C	
PL/SQL	PHP	Java	PHP	Perl
	C/C++	PHP	Python	
	VB	Ruby	JavaScript	

注：

1) 部分语言在使用的过程中常会组合使用，如果支持 JAVA 的同时不支持 JSP 或者 JS，则不宜用于 JAVA 网站类项目。

2) 部分静态源代码安全分析工具需要单独购买 License 来支持新增开发语言，在选购安全检测工具时宜考虑厂家销售策略，可以对软件支持的编程语言、增加新类型编程语言所需要的预算进行充分的考虑。

3.2 支持扫描编译后的代码

软件越来越倾向于组件化开发的模式，应用程序通常会包含大量第三方程序的库，而这些库并没有提供代码。支持对编译后的二进制或字节码进行扫描将会发现更多的潜在安全问题。

3.3 支持的扫描环境

静态源代码安全分析工具应支持业界主流开发环境（主要指操作系统），包括但不限于：

- LINUX
- WINDOWS
- Mac OS X
- UNIX

考虑到静态源代码安全分析工具在不同平台下的安装文件可能不同。在选择静态源代码安全分析工具时候需要特别注意。比如：

- LINUX 下的安装介质与 AIX 环境下的安装介质可能会不同；
- 某类环境下不同操作系统版本要求的安装介质也可能不同。例如 Mac OS 不同版本可能需要不同的安装介质。

同时应考虑到现阶段服务器市场的发展趋势和操作系统的成本因素，工具宜支持 Linux 等开源操作系统。因此，在需要对多种平台进行源代码安全分析的情况下，应重点考虑该工具对不同系统和平台的支持能力。

3.4 支持的安全漏洞类型

安全漏洞是信息系统在开发生命周期的各个阶段中产生的某类问题，这些问题会对系统的安全（机密性、完整性、可用性）产生影响。各个漏洞数据库和索

引（如 CVE、CNVD）收录了大量已知的安全漏洞，但即便是最全的漏洞库也不可能收录所有的漏洞。静态源代码安全分析工具应涵盖设计、实现过现中出现的常见重要安全问题。安全漏洞有很多种分类方式，OWASP TOP10 是业界被普遍认可的一种基于技术类型的分类方式。静态源代码安全分析工具应覆盖 OWASP TOP10 所包含的危害性最大或者攻击成本最低的漏洞，如下所示：

1）注入类（例如：SQL 注入、XML 注入等）；

2）失效的身份认证和会话管理；

3）跨站脚本（XSS）；

4）不安全的直接对象引用；

5）安全配置错误；

6）敏感信息泄漏；

7）功能级访问控制缺失；

8）跨站请求伪造（CSRF）；

9）使用含有已知漏洞的组件；

10）未验证的重定向和转发。

注：

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

3.5 支持快速定位问题的功能

在人工审计安全问题的过程中，经常需要由问题的触发点追溯到上一个变量或者文件。为了准确定位及提高审计效率，工具的用户界面宜具备搜索功能，以及不同文件间跳转的功能。搜索功能包括但不限于：

- 关键字

- 类
- 函数
- 文件

3.6 支持分析结果的对比分析

在静态源代码安全分析过程中，大部分程序不会一次通过，可能经过多次反复的修改和检测。在修改原问题时可能引入新的问题。在问题量巨大的情况下，静态源代码安全检测工具宜提供前后两个版本分析出的缺陷的对比功能。

3.7 支持软件安全等级的评定和安全趋势的分析

常规分析结果应包含缺陷的类型、数量等信息。但仅凭该信息，不能对产品安全等级有比较清晰的认识。宜参考数量、严重性和企业自身系统的安全等级等因素，通过配置，对检测结果给出等级评定，并给出其趋势分析图，从宏观的角度反映组织整体的安全开发状况，方便向组织级进行汇报或展示。

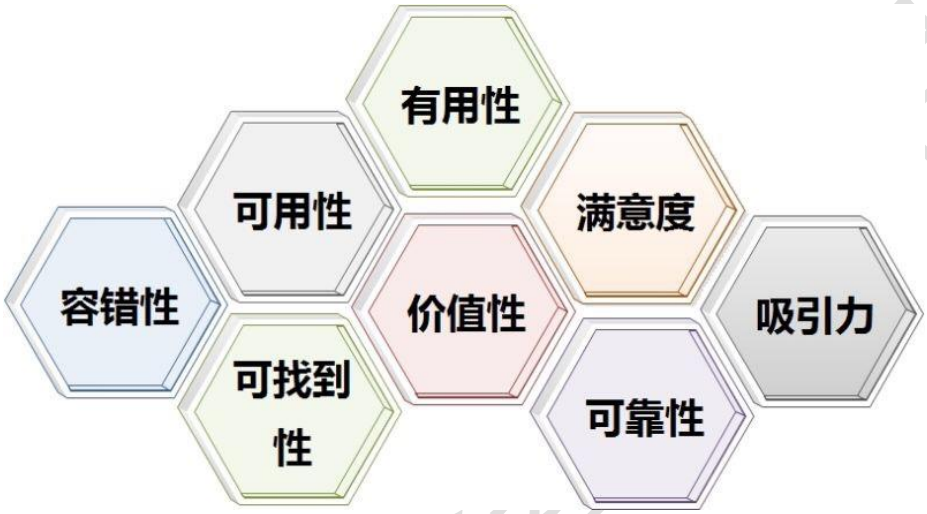
3.8 支持提供漏洞修复指导

普通用户很难完全知晓静态源代码安全分析工具所发现安全漏洞的具体含义。因此，静态源代码安全分析工具应对每一类缺陷进行必要的解释，并提供较明确的修复建议，便于用户理解和修复该安全问题。在选择静态源代码安全分析工具时，可对比每种工具对安全漏洞的“解释说明”和“修复建议”，选择“解释说明”和“修复建议”的内容更易于理解和接受，更具有针对性的静态源代码安全分析工具。部分工具还提供从攻击入口到问题出发点的跟踪流，这些辅助功能将对安全缺陷的确认和修复起到比较好的帮助作用。

4 用户体验

通俗来讲就是“这个东西好不好用，用起来方不方便”。因此，用户体验是主观的，且其注重实际应用。

图 2.1 用户体验特性蜂窝图



其主要特性详细解释如下表：

表 2.2 用户体验特性

序号	特性	概述
1	可用性	表示设计的产品功能应当很好地满足用户需求。为用户设定清晰的目标，符合用户的需要，同时告诉用户产品能够完成什么事。静态源代码安全测试工具应避免出现扫描卡死或异常退出等导致可用性差的缺陷。
2	有用性	表示设计的产品应当是有用的，面对的用户需求是真实的。而不应当设计对用户毫无用处的功能。静态源代码安全测试工具应能够发现业界通用的、典型的漏洞，帮助用户提升产品安全质量。

3	可找到性	应当提供良好的导航和定位元素，使用户能很快的找到所需信息或操作按钮等。静态源代码安全测试工具应满足客户可找到性的最大化需求，测试工具应设置合理的菜单深度、位置和提示信息，符合常用菜单界面的设计风格，可以方便的找到所需要的功能。同时，易于学习和使用、减轻记忆负担，帮助用户高效工作，避免引起操作错误。通过简单地培训即可使用，方便在用户企业近一步推广。
4	满意度	是指软件产品元素应当满足用户的各种情感体验，这个是来源于情感设计。静态源代码安全测试工具应满足客户满意度的最大化需求，执行过程中提供及时反馈，让用户感觉到离目标还有多远。通过最小化、完成提示、多语种支持等交互式设计来提高用户的满意度体验。
5	可靠性	是指软件及运行结果能够让用户所信赖的，要尽量设计和提供使用户充分信赖的组件。表现为静态源代码安全检测工具无故障地执行指定功能的能力或可能性。静态源代码安全测试工具应满足客户可靠性的最大化需求，测试结果真实、可靠，不会因为客观原因而造成无法产生测试结果，或两次测试结果不一致的情况。
6	价值性	它是指使用户获得利益，而对于非盈利性的目标，也要能促使实现预期目标。静态源代码安全测试工具应满足客户价值性的最大化需求，通过使用静态扫描工具可以降低被检测对象的问题修复成本，提升软件质量和市场认可度等，给客户带来价值。
7	容错性	源代码安全检测工具应具备防止用户犯错，以及恢复错误的能力。静态源代码安全测试工具应满足客户容错性的最大化需求，测试工具应能够在客户操作错误后返回错误提示或者提供回退、终止的功能，保障安全检测工作的顺利进行。

8	吸引力	静态源代码安全测试工具应满足客户创造力和自我实现的最大化需求，测试工具应能够允许客户化自主的设定扫描配置、变更规则和检索条件，允许发现，让用户知道总有新的东西，避免枯燥感。提供给用户学习、应用的乐趣。

5 使用模式

因为软件开发流程与环境的不同，所以工具的使用模式也会不同。静态源代码安全分析工具宜支持客户对不同使用模式的需求。使用模式包括操作方式和访问方式，如下表：

表 2.3 使用模式类型表

使用模式		概述
操作方式	图形用户界面模式	图形用户界面更加直观易用，通过图形化的界面完成代码获取、检测、审计、生成报告和规则调整等功能。
	命令行模式	命令行模式需要学习命令才能操作，相比图形用户界面操作起来复杂。但命令行模式执行效率更高、灵活性更强便于其他工具的调用，是对专业工具使用人员很重要的操作方式。
	IDE 集成模式	静态源代码安全分析工具宜作为插件支持主流开发环境。开发人员进行开发过程中可方便的进行自我测试，尽快解决问题。如：Eclipse、Visual Studio 等。
访问方式	B/S 模式（即	其优点是统一了客户端，用户不必做特别的

	Browser/Server，浏览器/服务器模式）	安装，系统主要依赖于服务器端。宜支持各类浏览器兼容。
	C/S 模式（即 Client/Server,客户机和服务器模式）	其优点是可以把工作分配给客户端和服务端处理，发挥两端的处理能力。
	单机模式（即扫描软件安装在单台机器上）	集中化管理，利于对源码安全性进行管控。

6 扫描能力

6.1 基础扫描能力

1） 静态源代码安全分析工具应具有如下基础扫描能力：

- 静态代码安全分析工具应对代码本身做彻底和一致的检查，不去关心开发人员是否已经具备的安全意识及其“有关的”代码，也不应了解哪些代码易于通过动态黑盒测试来检测；
- 静态源代码安全分析工具应能指出安全问题的根源，而不仅仅是某种症状，需要为确认漏洞修复提供更好的依据；
- 静态源代码安全分析工具需要结合整个软件开发生命周期，在开发阶段就应方便开发人员使用及早发现错误，使开发人员有机会修正他们之前没有注意到而很有可能会发生的错误，实现边开发边测试。工具同时应充当安全编码知识转移的手段。

2） 静态源代码安全分析工具常见的检查情景：

- 类型检查

类型检查消除将消除全部类别的编程错误，例如检测出程序员将整型值赋予对象变量，通过编译时捕获错误，类型检查将防止运行错误类型检查具有一定的局限性就是会产生误报和漏报的问题。

- 风格检查

风格检查侧重于表面的东西，与程序执行的规则与空格、命名、否决函数、注释、程序结构等相关，程序员的风格各有不同，工具需要检测出那些影响代码的可读性和可维护性的错误，某些编译器已经可以实现风格检查功能。同时就要求我们应通过自定义规则的方法满足不同风格。

- 程序理解

程序理解可以帮助客户理解代码库中大量的代码，集成的 IDE 都会包含一些程序理解功能。除此之外更应该具备更高级的功能支持自动化进行程序重新分解组合的功能，例如对变量重命名，或者将一个函数分解为多个函数。

- 属性检查

工具对描述部分程序行为的部分规格说明进行检查，多数属性检查应通过应用逻辑分析，或者通过执行模型检查来显示其功能。而且属性检查的重点是“临时性安全属性”，它规定了一系列有序的事件，比如“一个内存位置，当释放后就不应该被读取”，多数属性检查工具都许可程序员编写自己的规格说明来检查程序特定的属性。当工具发现代码与规格说明不匹配的时候，会将其发现报告给用户。如下图：内存泄露，第一次调用 malloc()后，后续的 2 次对该函数的调用失败，那么在第一次调用中分配的内存就会发生泄露（永远应该释放那些分配过的内存）

```
        inBuf = (char*)malloc(bufSz);
        if(inBuf == NULL)
        return -1;
        outBuf=(char*)malloc(bufSz);
        if(outBuf==NULL)
           return -1;
```

- BUG 查找

BUG 查找是指出程序以程序员设想之外的方式运行的一些地方，可以预先准备一套 BUG 规则，来描述代码中通常可能存在 BUG 的一些模式。如下所示： 双检锁的目的是减少同步，确保永远只分配一个对象，但是这种惯用法并不起作用。

```
        if(this.fitz == NULL){
           synchronized (this) {
             if(this.fitz == NULL) {
                 this.fitz = new Fitz();
             }
          }
        }
```

- 安全审查

静态源代码安全分析工具往往更像是一种属性检查和 BUG 查找的混合体，许多安全属性可以直接的表达为程序属性。

- 行业标准支持

静态源代码安全分析工具应该支持主流的相关行业标准：

表 2.4 主流安全标准

序号	检查情景	概述
1	OWASP (Open Web Application Security Project)	OWASP 是一个开源的、非盈利的全球性安全组织，致力于应用软件的安全研究。我们的使命是使应用软件更加安全，使企业和组织能够对应用安全风险作出更清晰的决策。目前 OWASP 全球拥有 140 个分会近四万名会员，共同推动了安全标准、安全测试工具、安全指导手册等应用安全技术的发展。
2	CWE(Common Weakness Enumeration)	CWE 常见缺陷列表(Common Weakness Enumeration)是 MITRE（一个非盈利机构）继 CVE（Common Vulnerabilities and Exposures）之后的又一个安全漏洞词典。通过这一词典提供识别、减轻、阻止软件缺陷的通用标准。
3	CNVD	国家信息安全漏洞共享平台（China National Vulnerability Database，简称 CNVD）是由国家计算机网络应急技术处理协调中心(中文简称国家互联应急中心，英文简称 CNCERT）联合国内重要信息系统单位、基础电信运营商、网络安全厂商、软件厂商和互联网企业建立的信息安全漏洞信息共享知识库。
4	PCI(Payment Card Industry Data Security Standard)	第三方支付行业(支付卡行业 PCI DSS)数据安全标准，是由 PCI 安全标准委员会的创始成员(visa、mastercard、American Express、Discover Financial Services、JCB 等)制定，力在使国际上采用一致的数据安全措施，简称 PCI DSS。
5	SANS 20 Critical Security Controls	网络安全培训、认证组织 SANS 发布的前20类关键的安全缺陷。

6.2 扫描速度

扫描速度是用户比较关心的一个指标，特别是对于并行开发项目较多或者开发项目代码量非常庞大的企业用户。针对扫描速度的测评需要选择不同语言与大小的软件进行扫描对比。

如下列的计划：

序号	项目名称	开发语言	代码行数(万)	扫描时长
1	项目 1	C	10	
2	项目 2	JAVA	10	
3	项目 3	C#	10	
4	项目 4	Python	10	
5	项目 5	C	100	
6	项目 6	JAVA	100	
7	项目 7	C#	100	
8	项目 8	Python	100	
9	项目 6	JAVA	200	

除单个项目扫描之外，同时需要测试并发扫描的效率。测试时记录下 CPU 与内存的利用率以观察其引擎的性能（关于引擎的定义和分类请参考本章 6.4）。

6.3 扫描配置能力

为了更好的适应开发活动，提升工具易用性。扫描配置宜包含如下的能力：

序号	能力	概述
1	定时扫描	代码扫描有时会安排在代码版本构建完成之后。有时为了考虑硬件资源使用上的分配，也需要配置定时扫描。
2	查看实时扫描进度	一些代码量比较的大项目可能需要通过几个小时或者更长的时间才能完成扫描，因此扫描人员需要能看到真实的扫描进度。
3	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，逐个排队等待将会非常耗时，并发扫描能够最大化利用硬件的资源。并发扫描同时也是多用户时使用必须要支持的能力。
4	增量扫描	增量扫描通常应用于较大的项目，通过只扫描修改的部分达到减少扫描时间的目的。

6.4 支持的分析引擎

分析引擎指控制一定范围内所有功能的主程序。同时也代表着独特的发现问题的方法集合或者发现问题的角度。通过引擎将源码和安全分析规则对应起来，发现安全问题。

1） 应支持的分析引擎：

- 语义分析：分析程序中不安全的函数,方法的使用的安全问题。
- 配置分析：分析项目配置文件中的敏感信息和配置缺失的安全问题。

- 数据流分析：跟踪,记录并分析程序中的数据传递过程的安全问题。

2） 宜支持的分析引擎：

- 控制流分析：分析程序特定时间,状态下执行操作指令的安全问题。
- 结构分析：分析程序上下文环境,结构中的安全问题。

特定的引擎可更直观、高效、准确的发现特定类型缺陷。另外，各检测工具也会有其自身的特点、优势和专利技术，在评估分析引擎时，也需要考虑各检测工具自身的特点、优势和专利技术。它们与上述几种分析引擎相结合，提高发现安全问题的全面性。

6.5 漏报率

漏报率是考核工具扫描能力的重要指标。在评估工具之时，通常会选择一些包含典型漏洞的开源项目作扫描，或者预埋漏洞的软件进行扫描效果评估。

6.6 误报率

误报是指工具产生太多的无用信息，即当程序中实际不存在安全问题时，而报告了问题。大量的误报确实会带来很多困难，审查冗长的误报列表不仅仅是耗费审查人员大量的时间和精力，而且不得不查看列表中还可能遗漏掉隐藏在列表中的重要实际的漏洞。漏报的结果是需要为代码中存在安全漏洞的事实承担安全风险。

7 报表能力

报表能力对客户最显而易见的功能。工具应该提供多种不同的结果呈现方式。

7.1 支持基于角色的报表

工具应该有能力至少提供如下类型的报表：

1) 管理人员

- 汇总报表：提供高级别扫描结果汇总，即数据的汇总报表。
- 合规性报告：工具应该提供合规性报告以便于用户快速确定是否满足特定的行业规范。下列是一些潜在的行业规范：
  - OWASP TOP 10
  - CWE/SANS TOP 25
  - SarBanes-Oxley(SOX)
  - Payment Card Industry Data Security Standard (PCI DSS)
  - Basel ||

2) 技术人员

- 扫描结果细节报表：提供所有的技术细节让开发人员能够理解漏洞并知道出错的代码。细节报告至少应该包含如下的内容：
  - 漏洞汇总，漏洞的种类、个数及安全级别。
  - 出错的代码行、文件路径、文件名及代码的调用关系。
  - 为每个漏洞定制的有针对性的修复意见。

7.2 报表格式

工具应当支持尽量多的报表格式，如 PDF、HTML、Word、EXCEL、RTF 及 XML。

8 扩展能力

8.1 与第三方工具集成能力

评估与第三方的配套工具的集成能力。常见工具包括源码管理系统、缺陷跟踪系统、持续集成系统。除了能与主流的管理工具集成外，工具是否提供接口供任何第三方系统集成也是考核项之一。

8.2 与源码管理系统集成能力

静态源代码安全分析工具宜支持与代码版本管理工具协同自动化工作。通过与版本管理工具如 SVN 集成，在不经由人为操作的情况下由机器按照预先设定的时间/版本变更完成代码获取、检测。自动化是节省人力、时间，提升测试覆盖率和质量的重要手段。以与源码管理系统集成的方式获取源码更符合安全性的要求。反之，源代码手工上传会有代码泄露的风险。

8.3 与缺陷跟踪系统集成能力

为了便于任务指派及更好的跟踪漏洞修复情况，工具宜提供与缺陷跟踪系统集成的接口。

常见的集成方式包括：

- 单个漏洞提交：在静态工具界面上填写表单（包括漏洞相关者、风险级别、备注等）提交；
- 批量导入：批量把某个项目的扫描结果提交到指定的缺陷管理系统。

8.4 与持续集成系统集成能力

对于一些定期发布版本的项目，自动化扫描流程非常必要，因此工具宜提供接口（如命令行发起扫描任务）与持续集成系统（如 Jenkins）结合。

接口考核点包括：

- 支持的源码管理系统有哪些？
- 是否支持权限管理？安全性如何保障？
- 是否支持在持续集成系统里显示汇总报表？

8.5 规则自定义能力

评估工具是否支持规则修改与自定义。此项将考察工具是否开放自定义规则的接口，是否易于掌握语法规则，以及可实现的程度。

8.5.1 修改漏洞规则能力

当某个漏洞检测出现误报或漏洞报时，工具应当允许客户对规则进行修改以保证精确度。

下列是常见规则修改的原因：

- 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。

一个扩展性好的工具应当支持如上遇到的修改规则需求。一部分工具仅支持通过界面配置的方式增加特定的输入输出方法以及过滤方法，而比较大的规则改动只能通过规则代码编写的方式完成。

8.5.2 新增漏洞规则能力

虽然静态源代码安全分析工具通用性较好，但各公司都有其自身业务特点。在默认情况下，一些个性化问题不能检测出来，或者检测出来在该业务环境下不存在的漏洞。因此，新增基于业务规则相关的漏洞往往用处很大。

新增漏洞检测规则的方式：

- **界面配置**

  部分静态工具提供以界面配置选择的方式组件各种条件来生成新的漏洞规则。这种方式的优点是容易操作、但灵活度不够。

- **编码**

  部分静态工具提供以编码的方式创建新的漏洞规则。编码语言如果是自定义语言，工具应当提供详细的 API 说明文档及教程。

8.6 与黑盒安全检测工具的配合能力

静态源代码安全分析不需要运行程序，发现的代码安全问题不能在实际运行状态下进行验证。如果代码级的静态源代码安全分级工具可以与黑盒安全测试工具进行相互验证，会提升确认安全问题的准确性、降低确认问题的工作量。

8.7 客户化能力

结合企业自身业务、流程和组织特点进行客户化定制开发后的工具能够更好的融入到该企业的环境中。

8.7.1 界面客户化

常见界面客户化的内容项包括：

- 是否支持整体替换；
- 是否支持替换 LOGO 及厂商名称；
- 是否支持调整界面布局；
- 是否支持修改界面内容；
- 是否支持替换显示语言。

8.7.2 功能客户化

部分用户可能会提出定制开发的需求，以满足在特定的环境中使用。常见的功能客户化的需求包括：

- 支持新的扫描语言；
- 支持新的开发框架；
- 扫描流程定制；
- 与特定客户系统的集成。

8.7.3 报表自定义

评估报表客户化的程度。理想的报表自定义提供可让客户修改内容与样式的模板。

报表自定义应该包括：

- 允许更改报表中显示的 LOGO；
- 允许修改字体、标题、页头、页脚、图表等界面样式；

- 允许添加漏洞注释信息；
- 允许标注哪些漏洞是误报，并可选择是否包含在报表中；
- 报表数据可通过自定义数据源的方式修改。

第三章 附录

附录一：静态源代码安全分析工具非技术能力基准

1 静态源代码安全分析工具采购评估依据

每个组织或企业都有其独特的环境或开发技术，评估好坏需要结合企业自身的需求来考虑，符合需求才是关键。一些需要在评估过程中考虑的常见要点如下：

- 组织所采用的编码语言列表；
- 组织所采用的框架和库；
- 谁负责去执行扫描；
- 工具如何在软件开发生命周期中集成；
- 对于软件与硬件的采购预算；
- 需要使用工具的人数；
- 是否允许在外部提交扫描代码。

2 软件供应商公司实力评估依据

软件供应商总体实力是指公司的财务健康，人员素质及数量，公司总体的战略方向，对静态源代码扫描产品市场持续投入的可性行等方面的表现。

除了以上常见指标外，软件供应商在国内外市场的整体和领域内销售情况，软件供应商在领域内产品的发展路线，软件供应商的销售和技术支持体系，代理商体系，网上社区情况也是重要的审查内容。

实力雄厚的软件供应商可为产品的未来发展以及技术支持相关服务提供保障。

3 价格评估依据

产品的价格包括：

- 产品本身的价格
- 产品交付服务和培训的价格
- 产品维保和升级的价格

产品的计价模式取决于 License 管理策略，不同静态源代码安全分析工具的 License 管理策略各有不同。常见的 License 管理策略如下表所示：

表 3.1 常见 License 管理策略

License 管理策略	价格影响	评估标准
单次扫描允许的行数	允许单次扫描的行数越多，价格越贵。	根据公司采购预算及项目大小决定。
单套软件允许的并发数	并发扫描项目数越高，价格越贵。	根据公司采购预算及项目数量决定。
允许登录的用户数	基于云服务的软件通常以用户个数计价；还有可能按角色划分不同的价格。	根据公司采购预算及项目数量决定。
包含的模块数	模块包括核心模块和扩展模块。如果用户没有购买扩展模块将不会影响核心模块的使用。	根据公司采购预算及需求决定。
支持的扫描语言	支持的扫描语言越多，价格越高。	根据公司采购预算及使用语言决定。
License 个数	单机部署的桌面软件通常会按照	根据公司采购预算及自

	License 个数销售。	身使用需求决定。
使用时间	按照使用时间计算的收费方式。	根据公司采购预算及自身使用需求决定。
共享	对同一软件供应商的多款软件采用一个 License 共享使用。	根据公司采购预算及对同一个软件供应商其它产品的采购情况决定。

4 售前支持服务

售前支持服务对于论证软件的适用性起到支撑和协助的作用。评估此项内容有利于确保使用软件工具的企业快速准确的完成 POC 过程。

售前支持包括售前技术支持和培训介绍服务。技术支持部分通常包括安装部署、不同类型项目的扫描、产品功能调试。培训介绍服务一般包括产品功能演示讲解、扫描漏洞结果分析与报表生成。

5 售后支持服务

产品售后服务的力度可决定用户是否能很好的使用产品。因为静态源代码安全分析工具是专业性较强的产品，因此用户往往需要花费很长的时间才能够完全掌握使用。并且，把静态源代码安全分析工具融入到现有的软件开发流程中也需要售后服务的有力支撑。售后支持服务包含售后交付、培训及长期的售后服务方案。

售后交付能力需要考核交付方式及实施人员的资质、软件安全生命周期技术咨询的能力。售后培训包括产品使用培训与附加的安全相关培训（如：漏洞修复）。将统计培训的时长、周期与质量。

厂商的售后服务方案应包含维护内容、维护方式及响应时间。维护内容应涉及到纠错性维护、适应性维护和完善性维护。维护方式通常包括电话支持、邮件支持及本地支持。

6 产品更新能力

开发技术的更迭及新漏洞不断涌现使得安全测试产品自身的更新尤其重要。因此，用户在选购工具时，需要了解产品在未来 3-5 年的发展路线，产品更新和规则库更新的方式与周期。

附录二：引用

- List of Analyzers For Static Code Analysis (http://en.wikipedia.org/wiki/List_of_analyzers_for_static_code_analysis)
- Static Program Analysis (http://en.wikipedia.org/wiki/Static_program_analysis)
- OWASP Top 10 (http://www.owasp.org.cn/owasp-project/2013top10)
- CWE (https://cwe.mitre.org/data/slices/2000.html)

附录三：ASC 应用安全联盟介绍

1 联盟简介

应用安全联盟是由行业专家、应用安全厂商、应用安全行业用户的代表共同成立的认同开源安 全技 术以及最佳实践安全标准的非盈利组织。 作为一个专业的应用安全群体，我们致力于建立行业应用安全基准、促进行业技 术交流，为厂商、用户、安全从业人员提供技术指导。

应用安全联盟成员包括行业专家、应用安全厂商和应用安全行业客户。

2 联盟章程

我们的使命：

我们致力于建立行业应用安全基准、研究最新安全技术、促进行业技术交流，为 厂商、用户、安全从业人员提供技术指导，推动中国业务及应用安全技术发展。

我们做什么

- 创建一个开放的应用安全群体，持续研究、讨论、共享应用安全技术
- 在全国范围内普及应用安全技术
- 建立一个中立的应用安全技术研究机构

我们不主张

- 不提供特定供应商的技术、服务和解决方案
- 不代表任何公司立场，仅推动应用安全技术发展。

3 联盟项目

- 应用安全联盟的各个项目，主要在于建立应用安全行业的各类基准，帮助行业厂商改进产品，帮助行业用户更好的选择产品。
- 应用安全联盟成员自愿参与并完善应用安全联盟的各个项目，共同推动应用安全领域技术发展。
- 项目包括：

  Web 扫描器评估基准

  Web 应用防火墙测评基准

  Web 安全威胁分类基准

  代码审核工具测评基准

  数据库审计系统测评基准

  移动应用安全检测基准

4 联系我们

姓 名： 张小姐

电 话： 4000-983-183

E-mail： service@seczone.org

静态源代码安全分析工具测评基准（第一版）

应用安全联盟的各个项目，主要在于建立应用安全行业的各类基准，帮助行业厂商改进产品，帮助行业用户更好的选择产品。

应用安全联盟成员自愿参与并完善应用安全联盟的各个项目，共同推动应用安全领域技术发展。

ASC项目-基准系列

- Web扫描器评估基准
- Web应用防火墙测评基准
- Web安全威胁分类基准
- **代码审核工具测评基准**
- 数据库审计系统测评基准
- 移动应用安全检测基准

电话：4000-983-183
邮箱：asc@seczone.org
网址：www.seczone.org