HP Fortify SCA
部署指南

在部署HP Fortify Static Code Analyzer（SCA）时，企业可能会遇到一些问题，导致他们延误部署或者无法部署该解决方案。HP Fortify Professional Services提供了以下“最佳做法”指南来帮助企业成功地部署基于SCA的解决方案。

1）你是在部署解决方案，而不是工具

这是很多技术企业认为最具挑战性的障碍。我们都知道，电锯是一种工具，它可以将树木锯成木材；但你不会直接开始使用电锯或者让别人使用它，而是需要一些准备工作和培训，考虑每次切割的目的，以及定期检查切割状况。如果你不这样做，你可能会看到锯条卡在树中无法使用；这会造成物件损坏或者让操作者严重受伤。总之，不会出现你预期的结果。

基于SCA的解决方案可以实现一个或多个预期的目标；通常情况下，即通过修复漏洞来降低业务风险。

SCA还可以用来改善代码质量、教育开发人员、降低成本，或者实现合规性。但其主要目的是降低业务风险。

通常一个解决方案包括人、技术，以及人和技术实现目标的过程。思考一下，谁负责修复SCA发现的漏洞。

- 没有修复代码的话，你将不会降低任何业务风险；
- 在代码修复之前，应该有人对代码进行分类，然后分别解决这些问题；
- 在对代码分类前，应该有人了解原始问题，以排除不相关的结果；
- 在了解原始问题之前，需要有人成功运行扫描，获取正确版本的源代码，这个人必须知道如何构建特定的应用；
- 最后，在开始扫描之前，需要有人决定或确定扫描哪个项目，更重要的是解决这一问题：“项目是什么？”

请注意你的人员将需要做的工作。SCA不会决定需要扫描哪些项目，SCA也不会修复你的漏洞。这个解决方案需要你的团队付出努力，在安装任何软件之前，应该提前规划这个工作流程。

2）成功的解决方案应以开发团队为中心

虽然企业很想建立这样的解决方案，即让安全团队可以在隔离环境中获取和扫描应用，但这是部署解决方案的错误方法。

原因包括：

- 安全人员通常无法构建软件应用，这通常需要开发项目的构建工程师所具备的的专业知识；
- 安全人员通常没有足够的技能，或者被授权来检查源代码中的变更，因此他们无法修复发现的漏洞；
- 不管安全人员是否能够理解在应用中发现的问题，他们通常都需要开发团队的成员来与他们一起审查漏洞。

正确的解决方案是对开发团队进行培训，让他们进行扫描。

3）从试点开发团队开始

从小处着手。以开发团队为中心建立解决方案；并让试点开发团队为该解决方案中的流程编写规范（例如“程序启动假设”；“主要工作流程步骤”；“备选措施”；“程序完成输出”）。

认真选择你的试点团队，这个团队应该是这样的：

- 小型。5个或5个以下开发人员的团队最容易开展工作；
- 备受尊敬。他们应该是同行团队中的佼佼者；
- 这个团队致力于的产品应该是企业取得成功至关重要的软件；
- 理想情况下，他们应该使用SCA支持的编程语言。例如，SCA支持Java、微软C#.NET和C/C++，并已经支持这些主流语言长达8年或更长时间。

按上述要求可以选出试点团队，最终其成功将在社交网络上传播开来，这可以让其他团队复制这种成功模式（必要的话，需要作出改进）。

4）正确确定推动该解决方案所需要做的工作

在每100到200名开发人员中，你大约需要一名全职专业人员来推动这个解决方案。这个人将帮助进行权限提升、管理定期系统维护和审计，以及监控定义的角色是否正确遵循流程。如果你不打算安排这样的职位，部署将无法成功。所以你应该预算好这笔费用，并指派具有必要技能的人员。如果你的企业没有这样的人，联系HP Fortify了解“常驻顾问”的费用。

5）在可能的情况下利用现有的技术

SCA给开发人员带来了新的任务。从试验来看，这似乎会增加他们的工作量；但鉴于漏洞生态系统的其他部分，在项目推出之前发现问题应该会逐渐减少工作量和企业压力。

企业引入新技术或工艺带来的问题是，这会迫使开发人员替换他们已经熟悉的系统。但企业最好不这么做。例如，如果你可以整合SCA与JIRA，但开发人员正在使用FogBugz，你不应该要求开发人员使用JIRA，而应该弄清楚如何整合漏洞追踪与FogBugz才对。

6）转换应侧重于零构建警告

在两种情况下我会听到这样的问题，“请你告诉我，如果你给机器输入了错误的数字，那么，是否还能得到正确的答案？”我无法正确领会这类想法。

——援引自Charles Babbage的《一个哲学家生涯的片段》

“无用输入无用输出”是数据系统建设者和设计者熟悉的一句话。这句话也适用于SCA转换——这是分析的第一步。

对于任何SCA构建模型，你可以使用“sourceanalyzer -b model_name -show-build-warnings”来查看这种转换是否检测出任何问题。此外，你可以使用以下命令来获取已转换文件的列表：
“sourceanalyzer -b model_name -show-files”

如果输出不如你的预期，你可以继续操作直到发现和纠正问题。

不要扫描不完整或存在错误的构建模型，这种扫描结果会浪费你的时间。

7）使用命令行来执行无错误扫描

不要使用“扫描向导”或任何图形应用来进行SCA扫描。你可能无法明白底层转换中使用了哪些参数以及为什么。其结果是，当出问题的时候，你不清楚如何解决这个问题。

这个规则的例外是微软Visual Studio .NET集成开发环境；在这种环境中，对于大多数SCA版本，图形版本的转换要比命令行更强大。当你在图形模式运行这种转换和扫描时，你会发现所使用的命令行参数，在隐藏的子目录Local Settings\Application Data\Fortify\VS*中的用户配置文件目录中。

8）建立安全检查点，只要可行即可

在试点团队展示了解决方案的运行后，应建立产品发布的检查点，这要求进行扫描并解决与安全政策相关的一些问题。这个政策应该记录在你的解决方案中。它应该包含类似这样的要求“修复所有在2015年1月1日后发现的问题”，并应该包含详细信息，例如谁应该负责扫描、哪些应用需要被扫描等。这个检查点应该只可用于开发团队，因为是他们将这种概念引入到过程中的。

9）先修复最严重的问题

这可能听起来很明显，当开发人员第一次审查SCA结果时，他们通常会倾向于先解决他/她了解的问题，而不是最严重的问题。企业可通过Fortify用户界面过滤器来获取指导和培训。

10）不要好高骛远

我们不可能解决所有问题，事实上，永远不可能。

请记住，你的开发人员都是聪明的工程师，如果他们在第一个问题（一个误报）花了7分钟，并且他们还有1万个问题需要解决，他们可能会认为他们将需要7万分钟，这意味着他们需要超过29个工作周的工作时间。你可能很快会发现他们开始“生病”或者整天在LinkedIn寻找新工作。

如果在初次扫描中发现很多问题，考虑使用过滤来显示最糟糕的真正的问题。另一种有用的方法是设置底线，例如指定在期限日期后“没有新问题”。这个期限日期之前的其他问题可以成为修复团队的单独的项目。

为了完成上述工作，你可能需要专门的技术人员来处理SCA结果、创建过滤器，并将其正确部署到服务器以及流程定义的相关步骤。如果你需要的话，你可以联系HP Fortify Professional Services获取有偿帮助。

11）避免“可利用性”陷阱

下面这个阴影部分节选自Brian Chess和Jacob West的《通过静态分析实现安全编程》（Addisson-Wesley出版社，2007年，版权所有）。

避免可利用性争辩

安全审查不应该是关于制造漏洞利用，但在很多时候，审查团队不得不开发漏洞利用。为了明白其中的原因，我们需要考虑在安全审查时一段代码可能获得的三种可能的裁定：

- 明显可利用代码
- 模糊代码
- 明显安全代码

这三者之间没有明显的分界线；它们属于一个范畴。只是两端要比中间的更好处理；明显可利用的代码需要修复，明显安全的代码可以置之不管。而中间的模糊代码最难处理。代码之所以模糊是因为其逻辑难以理解，因为我们难以确定在何种情况下代码会被调用，或者很难看到攻击者可能如何利用这个代码。

审查者对待模糊代码的方式可能会给企业带来风险。如果审查者在代码修复前需要证明代码的可利用性，审查者可能最后会犯错并忽略可利用的漏洞。当程序员说，“除非你能证明它是可利用的，否则我不会修复”，这里就是可利用性陷阱。（想了解程序员拒绝修复安全漏洞的更多理由，请参见侧面的“程序员拒绝修复糟糕代码的5个站不住脚的借口”）。

这个可利用性漏洞很危险，主要有两个原因：首先，开发漏洞利用很耗费时间。你开发漏洞利用的时间往往会超过寻找更多漏洞的时间。其次，开发漏洞利用是一项技能。如果你不知道开发漏洞利用呢？这是否意味着它无法被利用，还是说其实你根本不知道如何正确利用它？

不要落入可利用性陷阱：你应该修复漏洞！

如果一段代码不是明显安全的，你应该确保它成为明显安全的代码。这种时候可能需要冗余安全检查；或者这需要注释来提供可验证的方法以确定该代码的安全性。还有的时候，它会插入一个可利用的漏洞。当没有发现任何错误时，程序员并不会想要更改代码，因为任何变更都可能引入新的漏洞。但是推出包含漏洞的产品也不是好事。

被忽视的漏洞可能最终导致新的漏洞利用，而更糟糕的是，漏洞可能不需要具有可利用性就能对企业的声誉造成影响。例如，安全研究人员在发现新的缓冲区溢出后，可以通过发布这个信息来获取名利和荣耀，即使攻击者不可能围绕该漏洞构建攻击。这就是说，尽管所有迹象都表明这是无法被利用的漏洞，软件公司有时候也会发布安全补丁。

程序员拒绝修复糟糕代码的5个站不住脚的借口

对于安全检查中未被修复的漏洞，那些不了解软件安全的程序员会给出各种各样的理由。我们最常听到的理由是“我不认为这是可被利用的”。所有代码审查者都有自己的借口，下面是他们忽视安全问题最常用的似是而非的借口：

1. “我相信系统管理员。”
尽管我知道他们以前错误配置过软件，但我知道他们这次不会出错，所以我并不需要验证我的程序是否得到合理配置。

2. “你必须进行身份验证，然后才能访问该网页。”
攻击者怎么可能会获得用户名和密码？如果你有用户名和密码，根据定义，你就是好人，也就是说，你就不会攻击系统。

3. “没有人会想到这样做！”
用户手册非常清楚地指出，用户名不能超过26个字符，并且GUI可以防止你输入超过26个字符。那么，在我读取已保存的文件时，我为什么需要执行边界检查？

4. “那个函数调用不可能出问题。”
我已经在我的Windows桌面运行了100万次，当它在128处理器的Sun服务器运行时，怎么可能会出问题？

5. “我们没想过将这个代码用于生产环境。”
是的，我们知道这是已经连续数年出货的产品的一部分，但在编写这个代码时，我们没想过它会用于生产环境，所以你在审查代码时应该考虑到这一点。

了解更多信息请访问：
hp.com/go/fortifyservices