源代码审计综述*

向灵孜 / 信息安全共性技术国家工程研究中心

【摘　要】本文阐述了源代码审计原则，概述了源代码审计的方法和技术，对比了源代码审计工具，表明了源代码审计的价值和意义，最后对源代码审计技术的发展趋势做出预估。

【关键词】代码审计　安全编码　信息安全

1 引言

近年来，应用软件随着信息系统的发展，功能随之增大，相应的，程序源代码的规模也加大了，容易被利用的安全漏洞和代码后门也不再局限于以往，这就使得用传统软件测试方法来检测源代码中的安全漏洞非常困难。所以从源代码审计的角度，对源代码进行检测和分析，从而从根本上保护软件和信息系统的安全，杜绝了代码后门又能够避免潜在的漏洞安全威胁，进一步保障了信息安全。

在对应用软件和信息系统进行安全评估的时候，很多问题都是出现在软件的编码阶段，如SQL注入、XML实体攻击、XSS跨站脚本攻击等[1]。这些问题都需要通过修改软件的源代码来解决。因此，对于应用软件和信息系统的安全问题必须从底层的源代码抓起，在其上做到最佳防范。

本文阐述了源代码审计原则，概述了源代码审计的方法和技术，对比了源代码审计工具，表明了源代码审计的价值和意义，最后对源代码审计技术的发展趋势做出预估。

2 源代码审计原则

代码审计的过程如下，首先阅读目标系统（被审计）源代码，然后检查保证代码正确安全的控制是否设置在了关键的逻辑流位置上。它的主要目的是使得安全的开发技术被研发人员和编码人员严格遵循。通常来讲，如果被审计的目标系统经过一次完整的源代码安全审计之后，能够规避大部

*基金项目：国家“十二五”863项目“云计算安全体系架构研究”（项目编号：No. 2013AA01A214）。

分的应用程序的源代码漏洞，提高模拟渗透测试的精度。

2.1 所有的输入均不可信

在源代码程序中有很多的输入，包括系统内外部，由于用户的输入被应用程序处理，然后分析，进而展示，那每一个有可能被恶意利用或者造成代码纰漏的输入都是可疑的。据统计，在漏洞里面，由输入数据引起的安全问题高达90%，所以，对于程序代码中数据输入的格式与内容进行严格限定，能起到一定的安全控制作用[2]。

2.2 遵循安全编码规范

CERT是美国软件工程研究所（SEI）的计算机安全应急响应组，它编制了多种语言的安全编码规范，其中，Java安全编码标准已经于2011年9月14日出版发行（Addison-Wesley出版社）。该安全编码标准着重于Java程序中最容易出错、产生代码安全问题的环节，详细制定了编码层保障安全的规则，同时也提供了规避产生类似错误的最佳操作指南[3]。

Robert C. Seacord（CERT标准主编）提道，CERT Java安全编码标准是作为SEI安全编码项目的一部分。之前，CERT安全编码团队已经编制过很多的安全编码规则或标准，其中最著名的是CERT C安全编码标准（Addison-Wesley出版社2005）和CERT C++安全编码标准（Addison-Wesley出版社2008）。它们分别提供了C和C++编程语言中对于安全编码的指导方针和最佳操作指南，其目的是避免不安全的编码操作和可能会被利用的代码后门导致的潜在安全风险[4]。

遵循安全的编码规范在代码开发阶段很有必要，软件系统因为编码规范、安全，会更健壮，抵抗恶意攻击的能力也越强。源代码审计制定了相应技术的源代码安全规范，并在审查的过程中检查应用程序是否严格遵从现有的安全编码规范。若是没有遵循安全编码规范，但无造成漏洞风险，这样的源代码仍然存在安全风险，需要在最后的审计报告中明确指出。

2.3 漏洞与安全编码内在关系

现有的代码安全审计主要是查找传统或者已知代码的安全漏洞，这些安全漏洞主要是一些国际权威组织比如OWASP、CWE、CVE、SANS公布的代码安全漏洞，这些漏洞都是基于不同的安全漏洞模型来查找的，主要使用了数据流、控制流等技术查找恶意数据的入口点和恶意数据可能被利用的点（出口点），利用人工分析从入口点到出口点是否有风险来判断问题是否真实存在，并且所有的入口点及出口点都是基于开发源代码语言的API来查找的，与本身代码的逻辑无关[5]。

CWE是一个社区开发的常见软件弱点的正式列表。它是一种用来描述软件安全弱点的通用语言，一个安全软件用来定位漏洞的标准量尺，以及一个用于弱点识别、缓解和预防所使用的基线标准。CWE作为统一的标准，它无论在广度还是深度方面统一了这一领域最有价值的内容，平衡了学术界、商业界以及政府在思维和认识方面的不同。CWE的目标是促进代码审计行业的标准化和成熟化，同时大力推进各类组织对他们采购或开发的软件质量进行审查的能力[6]。

计算机安全应急响应组（Computer Emergency Response Team）是专门处理计算机网络安全问题的组织[7]。在1988年莫里斯蠕虫横扫互联网之后，在美国联邦政府资助下，卡耐基梅隆成立了第一个CERT组织。主要对常用的编程语言制定相应的安全编码规范，例如C，C++和Java等等。

CWE和CERT安全编码标准独立执行各自功能但是又相互支持着，简单来说，CWE提供了一个漏洞弱点的综合库，而CERT安全编码规范负责识别不安全的编码结构，如果在代码里面出现该结构，就可能暴露软件的弱点或漏洞。因为并不是CWE里面所有的弱点都能够在任意一种安全编码规范里面得到呈现，因为每一种语言不可能呈现每一种漏洞，而且CWE还包括一些高层设计问题。也并非所有的CERT安全编码规则都能够直接

映射CWE里面的漏洞，因为一些编码错误会以各种各样的方式呈现处理，这些方式可能不直接关联CWE里面给定的漏洞或弱点。总而言之，这两种工具在软件系统评估安全的过程中都是必需的。

2.4 反向思考

在源代码审计的过程中，需要审计人员站在一个恶意用户或者攻击者的角度上，保持思维的灵活连续，从各个方面对代码的所有问题进行排查，对于异常问题的检查，要考虑到其波及范围和漏洞影响。

3 源代码审计方法和技术概述

3.1 代码安全审计方法

代码安全审计的实施阶段，有一系列常用的方法：

（1）自上而下

由于代码审计存在于应用程序的生命周期之内，故自上而下的审计方法是当应用程序收到了用户请求，对其进行逻辑上的处理和操作，使得用户能够得到最终的返回结果[8]。自上而下代码审计法跟踪了所有的外来输入（包括用户输入或环境变量输入），凡是有可能被用户恶意控制的变量和容易对内部变量造成污染的函数或者方法都被严格跟踪。一旦参数被接受，就会顺着代码逻辑被遍历跟踪，一直到找到可能有安全威胁的代码或者直到所有的输入都被过滤或限定为安全为止。

（2）自下而上

该方法恰恰与自上而下相反，根据敏感函数的关键词字典，从应用点回溯器接受参数，一步一步向上跟踪，直到排除嫌疑或发现安全隐患为止。此方法需要审计人员对这些敏感函数的内部机理和使用方法非常了解，这样才能判断某些非法参数的输入是不会有安全风险。

（3）逻辑路径覆盖

逻辑路径覆盖是由生命周期和代码的逻辑上出发，人工或者使用工具来遍历代码逻辑上所有可能形成的路径，发现这些路径上可能存在的安全隐患。它是一种基于代码逻辑的能够发现难以在黑盒或灰盒测试中检查出来的漏洞的一种遍历测试方式。虽然能够帮助研发人员在编码阶段检测到易引发的逻辑上的危险漏洞，但是投入到逻辑覆盖路径上的人力成本很高，审计时间较长，产出投入比低。而通常情况下，代码逻辑安全问题通过一般常规的测试，显现出来的安全问题比较少，或者有部分隐蔽问题，难以被代码审计人员或工具发现。

3.2 代码安全审计技术

源代码安全分析其实早在1976年就有研究了，当时Colorado大学的Lloyd D. Fosdick和Leon J.Osterweil曾在*ACM Computing Surveys*上发表了著名的*Data Flow Analysis in Software Reliability*，文章中提到了常用的代码审计技术：边界检测、数据分类验证、状态机系统、边界检测、数据类型验证、控制流分析、数据流分析、状态机系统等[9]。

目前，软件源代码的安全性分析方法主要可分为静态分析与动态分析，国内外一些研究者在这一方面做了不少研究，提出一些不同的软件源代码的安全性分析实现方法，并设计了相关的应用系统。国际标准组织也提供了一系列的安全编码标准。

2007年James W. Moore和Robert C. Seacord在*Seacord Secure Coding Standards*一文中提到，由于编码漏洞导致的安全问题不断上升，ISO国际标准组织编写了一系列的安全编码以及安全使用编程语言避免漏洞的标准，如ISO/IEC TS 17961-2013 Information technology.Programming languages, their environments and system software interfaces. C secure coding rules；ISO/IEC 9899:2011 - Information technology—Programming languages – C等[10]。2010年开放式Web应用程序安全项目（OWASP）发布的安全编码指南里面虽不涉及编码安全规范的实施具体细节，但是提供了将编码规范转换成编码安全的具体要求，创建了安全编码标准并建立了一个可重用的对象库文件[11]。

2013年Robert C. Seacord等人编写了《C和C++安全编码》，结合国际标准C11和C++11及其最新发展，详细地阐述了C/C++语言及其相关库固有的安全问题和陷阱，系统性地总结了导致软件漏洞的各种常见编码错误，并给出了应对错误的解决方案[12]。同时也在*Secure Coding Rules: Past, Present, and Future*一文中对其源代码审计安全规则发展过程进行总结，并对未来源代码安全审计规则做出预测[13]。

2013年叶亮等人提出了基于安全规则的源代码分析方法[14]。2013年袁兵等人针对应用软件的恶意后门进行了代码审计，取得非常不错的效果[15]。2015年周诚等人提出了一种检测源代码安全漏洞的代码审查方法，该方法结合编码规则，高效检测出源代码的安全漏洞[16]。

虽然动态方法对代码的规模没有限制，可以对大型程序进行检测，但不足之处是检测的效果严重依赖输入方法，只有当特定的输入使代码执行到危险点时，漏洞才会被发现，所以这种方法漏报率较高。现在代码审计的主流技术是采用基于安全规则的源代码静态分析，包括数据流控制流分析、词法语法分析、基于抽象语法树的语义分析、规则检查分析等。它可以由人工进行，充分发挥人的逻辑思维优势，也可以借助软件工具自动进行。

下面介绍静态分析源代码的主要技术：

（1）数据流控制流分析

在静态环境下（即被测程序不运行条件下），从程序源代码中收集语义信息，检测使用操作中和对变量的定义中是否存在安全问题。数据流控制流分析主要是针对源程序代码上面的所有可能的逻辑路径和路径上所有变量的操作序列。

该方法有一定的缺陷，如信息量过大，容易造成组合爆炸等，且主要针对的是检测赋值引用异常及内存错误等，故往往作为其他检测方法的辅助方法使用。

（2）词法语法分析

在基本的语法词法分析上还采用了与CWE、CVE等安全漏洞库匹配的方法，这样能够检测出源代码中的不安全函数或漏洞个数和位置，该方法十分简单高效。但是由于缺少源代码的语义信息等，加上漏洞库里面已发现的漏洞数量是有限的，那些未公开的漏洞无法匹配检测，这样会提升误报率，降低了检测精度，所以往往是在代码审计的前期工作里面使用词法语法分析方法。

（3）基于抽象语法树的语义分析

源代码审计中的抽象语法树是通过扫描程序源代码来构建的，在去掉了冗余的语法结构后提取出源代码中的主要信息。通过抽象语法树来分析源代码的语义信息能够清晰地掌握源代码中的全局信息和模块信息甚至局部信息。这样使得隐藏于源代码中的危险漏洞被检测出来的概率很高，但是由于大多数安全漏洞比较复杂而且还存在很多0day漏洞，所以此方法的检测能力和准确率还有待提高。

（4）规则检查分析

规则检查分析针对的是源代码中存在的一些通用漏洞的规则，首先它采用了比较特定的语法来描述漏洞规则，然后用规则解析器来分析，将其分析结果转换成为系统能够识别的内部表达，同时采用图匹配和代码匹配来完成源代码的漏洞检测。在流程上，该方法比较完善，对于与安全编码规则相矛盾的源代码是很容易检测出来的。但是由于漏洞的特征不只是安全编码程序中所遵循的规则，所以只能够在已知的漏洞库中达到较好的检测效果，而对于0day漏洞，效果欠佳。

4 源代码审计工具

源代码安全性的分析离不开自动化的代码审计工具和与之结合的人工分析方法，源代码自动化审计工具先完成代码的漏洞扫描后，再由人工对漏洞进行分析、整改和确认。

4.1 工具简介

源代码分析技术随着计算机语言的不断演进而日趋完善。在不同的信息安全细分领域出现很

多好用的源代码分析产品，如Klocwork Insight、Coverity、Parasoft和Rational Software Analyzer等公司的产品，静态分析工具有Fortify公司的Fortify SCA、Armoire公司的Code Secure和Security Innovation公司的Checkmarx Suite等。

以下简单介绍几款主流源代码审计工具：

（1）Fortify SCA

它是一款用于测试软件源代码安全的工具，其内置了五大分析引擎：配置流、结构、控制流、数据流、语义。对源代码进行静态分析的过程中，按其特有的安全漏洞规则集进行查找和匹配，从而扫描出来源代码中存在的安全漏洞，并附有整理漏洞的报告。扫描的结果中包含了详细的漏洞信息和相关的安全知识说明，最后还提供了修复意见。

（2）FindBugs

它主要是针对采用Java语言来编写的源代码进行安全检查，检查Java里面的类或JAR文件。它通过对比字节码和一组缺陷模式来发现代码中可能存在的问题。它能够发现：多线程竞争、代码是否符合安全编码规范、性能等问题。

（3）PMD

PMD作为一种扫描效率高，功能强大的开源分析Java代码的工具，能够在不运行Java程序的情况下通过静态分析获知代码错误。而且PMD也附带了很多安全编码规则，通过匹配这些规则能够发现Java源代码中的许多问题。如潜在的bug、未使用的代码、资源关闭、复杂的表达式和重复的代码等。用户还可以自己定义规则来检测Java代码是否符合这些特定的编码规范。

（4）CheckStyle

CheckStyle主要是对代码规范进行检查，它属于SourceForge下的一个项目，是一个帮助编程人员来遵循某些安全编码规范的工具，实现了自动化代码规范检查过程。但是，它缺少增强代码质量和修复代码的功能。

4.2 工具对比

具体内容见表1。

5 源代码审计的价值与意义

源代码审计工作具有重要的应用价值，它一方面能够节约后期的安全投入，从源头上消除安全隐患，从根源上控制系统安全风险，有效减少了后期的安全评估、加固、维修补救等工作；另一方面，能够很大程度上降低系统安全风险、解决代码安全

表1　几种代码审计工具比较

工　具	目　的	检查项
Fortify SCA	检查JAVA，C/C++源代码是否匹配所有规则库中的漏洞特征	主要包括：扫描出约350种漏洞，并整理分类。分析的过程中与安全漏洞规则集进行全面匹配、查找，将源代码中存在的安全漏洞扫描出来，并给予整理报告。扫描结果提供详细的安全漏洞的信息、相关安全知识说明以及修复意见的提供
FindBugs	基于Bug Patterns概念，查找java byte code（.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Java源文件是否与代码规范相符、Javadoc注释、命名约定、标题、Import语句等	主要包括：Java doc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

隐患，从核心层面上来加强了整个安全保障体系的防护。

源代码审计工作意义在于提高应用软件源代码的质量、规避应用系统潜在后门带来的危害、防止信息系统的重要数据遭到泄露的同时又提升系统架构本身的安全性。避免被动防御处境，主动安全防御，明显节约了企业安全资金的投入，显著地提高了安全管理工作效率等。

6 源代码审计技术发展趋势

随着网络安全形势变得越来越复杂，安全问题日益增多，面对日新月异的攻击手段，很多安全操作必须从专业安全人员前移到系统管理人员，源代码审计亦是如此。近年来，虽然各大标准机构和厂商都提供了安全编码规则或者是安全编程标准，但是具体实施在源代码中的规则是有限的，如何实现源代码中自动进行安全规则检测或者是否遵循安全编码中最佳操作实践指南的建议，都需要利用人力和自动化工具进行探索和实现。在我国信息安全技术领域中，源代码审计以后的一种明显发展趋势是工具与服务一体化，其将漏洞（公共漏洞字典表CWE、2015OWASP十大Web漏洞以及设备、软件厂商公布的漏洞库）与安全编码规则相结合，并利用专业源代码静态分析工具对各种程序语言编写的源代码进行安全审计，定位源代码中存在的安全漏洞并分析漏洞风险，最终形成体系完整的审计流程，这一系列一体化服务将从根本上保护软件和信息系统的安全，杜绝了源代码后门，避免了潜在的漏洞安全威胁，进一步保障了我国信息安全。 END

参考文献：

[1] 秦晓军,甘水滔,陈左宁. 一种基于一阶逻辑的软件代码安全性缺陷静态检测技术[J].中国科学:信息科学,2014(01):108~129.

[2] 李俊.白盒代码安全审计方法浅析[J].牡丹江大学学报,2014(10):142~145.

[3] Fred Long, Dhruv Mohindra, Robert C.Seacord, Dean F.Sutherl. Java安全编码标准[M].北京:机械工业出版社, 2013:6~26.

[4] 王雷,陈归,金茂忠.基于约束分析与模型检测的代码安全漏洞检测方法研究[J].计算机研究与发展,2011(09):1659~1666.

[5] 梁业裕,徐坦,宁建创,杨明. 代码审计工作在整个安全保障体系中的重要价值[J].计算机安全,2013(12):32~35.

[6] 周宽久,郑红波,赖晓晨,刘春燕,迟宗正. 基于XML的软件安全静态检测方法研究[J].计算机工程与应用,2010(28):64~69.

[7] 孟云秀,赵正旭.基于源代码分析的软件静态测试[J].河北省科学院学报,2013(02):16~21.

[8] 方凯彬,闫巍. 移动互联网应用代码安全测试方法的使用[J].中国检验检疫,2013(11):31~32.

[9] LI Zhongyuan,QU Chengqin,ZHOU Xueguang, ZHUO Lifeng.Review of Public-Key Cryptosystem Based on the Error Correcting Code[J]. WuhanUniversity Journal of Natural Sciences,2014(06):489~496.

[10] James W. Moore,Robert C. Seacord.Secure Coding Standards[J].The Journal of Defense Software Engineering,2007,20(03):9~12.

[11] 2010年OWASP基金会.OWASP 安全编码规范快速参考指南[EB/OL].2010[2015-9-5].http://creativecommons.org/licenses/by-sa/3.0/.

[12] Robert C.Seacord.C和C++安全编码[M].北京:机械工业出版社, 2013.

[13] Robert C. Seacord.C Secure Coding Rules: Past, Present, and Future[J].informIT,2013,5(05):23~26.

[14] 叶亮. 基于安全规则的源代码分析方法研究[D].华中科技大学,2013.

[15] 袁兵,梁耿,黎祖锋,桂永宏,王睿. 恶意后门代码审计分析技术[J]. 计算机安全,2013(10):47~49.

[16] 周诚,张涛,马媛媛,李伟伟. 一种高效检测源代码安全漏洞的代码审查方法[J].现代电子技术,2015(05):83~86.