一种基于三维树模型的源代码安全缺陷分类方法

张 龑[1,2] 李舟军[1] 董国伟[3] 马殿富[1]
(北京航空航天大学计算机学院 北京 100191)[1] (湖北大学计算机与信息工程学院 武汉 430062)[2]
(中国信息安全测评中心 北京 100085)[3]

摘 要 提出了一种基于三维树模型的源代码缺陷分类方法,该方法综合考虑了缺陷产生的原因、造成的结果及其表现形式3方面的信息。实例分析结果表明:使用基于三维树模型的缺陷分类法得到的缺陷类别比CWE和Fortify中的缺陷分类更为精确和详细。该工作不仅有助于建立一种比较完善的源代码缺陷分类体系,而且对于缺陷检测规则的细化也具有实际的指导意义。

关键词 三维树模型,源代码,安全缺陷,分类法
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.5.014

Novel Taxonomy of Security Weakness in Source Code Based on Three-dimension Tree Model

ZHANG Yan[1,2] LI Zhou-jun[1] DONG Guo-wei[3] MA Dian-fu[1]
(School of Computer Science and Engineering, Beihang University, Beijing 100191, China)[1]
(School of Computer Science and Information Engineering, Hubei University, Wuhan 430062, China)[2]
(China Information Technology Security Evaluation Center, Beijing 100085, China)[3]

Abstract We presented a novel taxonomy of security weakness in source code based on three-dimension tree model, which synthetically considers the three aspects: the causes of the defect, the results and its form of expression. Case studies show that compared with CWE and Fortify, the taxonomy in this paper is more accurate and detailed. This paper is not only helpful to establish a kind of relatively complete source code defect classification system, but also very significant in practice to refine the rules of the security weakness detection.

Keywords Three-dimension tree model, Source code, Security weakness, Taxonomy

1 引言

绝大多数安全攻击都是由应用系统自身的漏洞引发的,而软件设计和开发过程中产生的缺陷是引发这些漏洞的主要原因。对软件源代码进行缺陷分析是减缓系统漏洞的一种有效方法,该方法可以比较全面地考虑和分析软件的执行路径,覆盖面广,并且能够实现自动化检测[1]。一般而言,源代码安全缺陷分析包括词法/语法分析、中间代码生成和缺陷检测3个步骤,如图1所示。首先,程序源代码通过词法和语法分析生成抽象语法树AST(Abstract Syntax Tree);然后,将AST转化为适用于缺陷检测的中间代码,如控制流图CFG(Control Flow Graph)、调用图CG(Call Graph)等;最后,根据缺陷检测规则利用各种静态分析技术对中间代码进行分析,发现并报告相应的缺陷。

缺陷检测规则构造是上述过程中的重要环节,而检测规则对于待检测的各类缺陷的描述能力直接影响到最终的分析结果。对缺陷按照自身的特征进行分类有助于缺陷检测规则的细化,提高缺陷分析的准确度,同时也有利于了解缺陷的本质及其产生的缘由,避免相同问题的反复出现,对预防与发现新的缺陷形式具有指导意义。

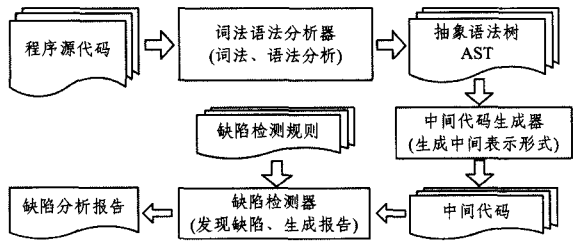

图1 源代码安全缺陷分析流程图

目前源代码缺陷分类方法的研究较少,但是系统漏洞分类方面的工作相对较多,这些分类法主要从成因[2-7]、威胁程度[8]、造成的影响[4,9-11]、攻击方式[5,6,11-15]、修复方式[10,16]、位置[4,5,17]等角度考虑漏洞的划分。虽然这些分类法覆盖面较广,但是不适用于源代码缺陷的分类,其主要原因在于这些方法多以某一特定漏洞的属性为分类标准,不能反映漏洞多方面的性质,并且其中涉及到源代码缺陷的内容较少。

本文首先分析了已有软件漏洞分类法的优缺点。在此基

到稿日期:2015-04-29 返修日期:2015-08-30 本文受国家自然科学基金(61170189,61370126,61202239),教育部博士点基金(20111102130003)资助。

张 龑(1974-),男,副教授,主要研究方向为信息安全,E-mail:zhangy@buaa.edu.cn;李舟军(1963-),男,教授,博士生导师,主要研究方向为网络与信息安全、数据挖掘与智能信息处理技术;董国伟(1981-),男,副研究员,主要研究方向为信息安全;马殿富(1960-),男,教授,博士生导师,主要研究方向为高可信软件技术、安全攸关系统。

础上，提出了一种基于三维树模型的源代码缺陷分类法，该分类法综合考虑了缺陷产生的原因、造成的结果及其表现形式等信息。最后，使用该方法对 CWE[18] 和 Fortify[19] 中描述的源代码安全缺陷进行了分类。从实际应用中发现，使用本文的方法可以建立一套比较完善的源代码缺陷分类体系，对预防和修复软件安全漏洞具有重要意义，同时对于缺陷检测规则的细化也具有指导意义。

本文第 2 节介绍了已有的软件漏洞分类方法；第 3 节给出了基于三维树模型的源代码缺陷分类方法；第 4 节通过实例说明了使用三维树模型缺陷分类法对 CWE 和 Fortify 中描述的缺陷进行分类的结果；最后对全文进行了总结。

2 软件漏洞分类法概述

现有的漏洞分类法基本上都是对环境、平台、配置、代码等支持应用系统运行的各个部分中漏洞的综合考虑。对于软件漏洞，从不同的角度有不同的定义，一般而言，比较普遍的有基于访问控制、基于状态空间、基于安全策略等定义形式[20]，由于需求的差异，现有分类法一般从产生的原因、造成的后果、严重程度、利用需求、平台等角度入手。

2.1 已有软件漏洞分类法介绍

针对 Unix 系统，T. Aslam 提出了基于产生原因的功能性错误分类法，将 Unix 的漏洞划分为 4 大类，分别为因设计错误、环境错误、编码错误和配置错误所引发的。其中设计错误主要包括需求分析和软件设计等过程中引入的错误；环境错误指由于操作环境限制而导致的错误，如编译器或操作系统缺陷导致的错误等；编码错误主要考虑了同步错误、条件验证错误等；配置错误主要涉及安装位置错误、安装参数错误、安装权限错误等内容[3]。

F. B. Cohen 提出了面向攻击方式的漏洞分类法，他对 100 多个可能的攻击集进行分析，提出了将软件漏洞划分为错误和遗漏、调用时的无用值、隐含的信任攻击、数据欺骗、过程旁路、分布式协同攻击、输入溢出、特洛伊木马、数据集成后导致错误、利用不完善的守护程序、利用系统中未公开的功能、攻击诱导的误操作、禁止审计、不断增加系统负载导致其失败、利用网络服务和协议、进程间通信攻击、竞争条件、不适当的缺省值等 18 类[12]。

I. Krsul 等人提出了面向影响的漏洞分类法，他们指出漏洞造成的影响可分为直接影响和间接影响两类，前者指当漏洞被利用时立即能觉察到的影响，后者指利用某漏洞后最终造成的影响。基于这一观点，他们将软件漏洞分为存取数据、执行命令、执行代码、拒绝服务等 4 大类[9]。

在基于多因素的漏洞分类法方面，C. E. Landwehr 等提出了基于漏洞的来源、时间和位置的分类法[5]，来源方面主要指木马、后门、逻辑炸弹等；时间方面指漏洞发生在软件生命周期的哪一阶段，如设计、编码、维护等；位置方面指漏洞发生在操作系统、支持软件等软件层面，还是硬件层面。在此基础上，K. Jiwnani 等人[4] 提出了基于原因、位置和影响的分类法，用于提炼软件开发阶段的问题，提高测试的针对性，其中原因方面主要考虑验证错误、域错误、顺序或别名错误等，位置方面主要考虑系统初始化、内存管理、进程管理或调度等，影响方面主要考虑未授权的存取、根或系统存取、拒绝服务等。

D. Wenliang 等人提出了基于漏洞生命周期的分类方法，他们将漏洞的生命周期定义为“引入-破坏-修复”的过程，根据引入原因、直接影响和修复方式对漏洞进行了分类。具体而言，引入原因方面包括输入验证错误、权限认证错误、顺序/别名错误等缺陷，直接影响方面包括非法执行代码、非法改变目标对象、非法访问目标资源、拒绝服务攻击等，修复方式方面包括实体虚假、实体缺失、实体错放、实体错误等[10]。

CWE(Common Weakness Enumeration)是美国 Mitre 公司推出的一个安全缺陷词典[18]，Mitre 希望通过它提供识别、减轻或阻止软件缺陷的通用标准，目前共收集了 800 多种缺陷。CWE 将缺陷分为 3 大类：代码缺陷类、环境缺陷类和配置缺陷类。其中代码缺陷类包括可执行代码缺陷、源代码缺陷和违反安全设计原则的缺陷 3 个子类。源代码缺陷又被划分为数据处理、API 误用、安全特征、时间和状态、错误处理、预示低劣的代码质量、通道和路径错误、处理器问题、习惯问题、Web 问题、用户接口错误、初始化和清理错误、指针问题、封装不充分等 14 个小类。不难看出，CWE 对源代码缺陷的分类主要基于产生的原因。

2.2 已有分类法在源代码缺陷分类方面的不足

从前面的分析中可以看出，目前针对源代码缺陷分类的研究较少，已有的软件漏洞分类法也不能直接应用于源代码缺陷分类中，这主要是因为：

1)深入考虑源代码缺陷分类方法的研究较少。在对漏洞进行分类时，源代码缺陷(或称编码错误)往往作为一个单独的类别存在[3-5,10]，而未对其进行更深层次的划分；在 CWE 中，也仅仅是基于缺陷产生的原因将源代码缺陷分为 14 类[18]。

2)某些分类法无法反映源代码层面的缺陷。对于面向攻击方式或影响的分类法，其基于的是漏洞被利用后引发的结果等因素[9,12]，没有表达这些问题与源代码缺陷之间的因果关系。

3)不能全面反映缺陷的信息。已有工作在分类时大都考虑单一的缺陷属性，如产生的原因、造成的结果等，不能全面反映缺陷多方面的性质，并且有些分类法中类别之间还有重叠的现象。这些都不利于源代码缺陷分析工作的开展。

3 基于三维树模型的源代码缺陷分类法

3.1 源代码缺陷分类的属性

为了规范源代码缺陷分析过程，提供更加完善详细的缺陷检测准则来提高分析的准确度，我们研究了源代码缺陷分类方法。由前面的分析可知，软件漏洞可以基于不同的漏洞属性进行分类，同样，源代码缺陷的分类也可以基于不同的缺陷属性。根据广泛的调研和实际经验发现，程序员经常使用以下 6 种属性来描述缺陷：(1)缺陷产生的内在原因，指在编写程序时源代码自身存在的问题，如编码时使用了危险函数而又未做必要的限定；(2)有意或无意的主观因素，有意缺陷是指开发者故意留下的缺陷，如逻辑炸弹、未声明的通道代码等；而无意缺陷指开发人员由于缺乏安全编程知识或只侧重功能与性能的实现，在编写程序时没有考虑代码自身的安全问题，导致了程序缺陷；(3)运行环境的外在因素，主要指与外部库函数或被调用代码之间接口上出现的问题，需要了解源代码运行的具体环境；(4)缺陷造成的结果，源代码缺陷在发

生错误后的直接结果，如发生缓冲区溢出等；(5)软件测试和使用时反馈的问题。软件测试和使用时能直接表现出来的出错特征，如 I/O 错误、计算错误、逻辑错误、数据加工错误、配置错误、操作系统及支持软件错误、接口错误、全局变量错误、预置数据库错误、程序崩溃等；(6)开发语言，某些缺陷出现在特定的开发语言中，如 J2EE 错误配置是 Java 语言特有的。

在上述 6 种属性的基础上，通过综合考虑缺陷的复杂性及分类方法的可扩展性，本文提出了针对源代码缺陷的 3 个分类属性，即缺陷产生的原因、缺陷造成的结果和缺陷的表现形式。

1)缺陷产生的原因。包括缺陷产生的内因、外因和主客观因素。已经归纳出了注入缺陷及验证缺失、应用程序编程接口错误、访问控制及密码失效、资源共享与竞争、异常处理、不安全的源代码质量、边界处理、配置错误、恶意代码等 9 类源代码缺陷产生的原因，如表 1 所列。

表 1　缺陷产生的原因

缺陷产生的原因(9 类)	有意	恶意代码	注入缺陷及验证缺失
	无意	环境无关	应用程序编程接口错误
			访问控制及密码失效
			资源共享及竞争
			异常处理
			不安全的源代码质量
			边界处理
		环境相关	配置错误

2)缺陷造成的结果。即缺陷被利用后产生的直接结果。已经分析并收集了注入、溢出、操纵、访问控制、泄露、文件系统、Web 攻击、死锁、拒绝服务等 9 类 35 个子类的缺陷结果，如表 2 所列。

表 2　缺陷造成的结果

缺陷造成的结果(9 类 35 个子类)	注入	SQL 注入
		命令注入
		XML 注入
		……
	溢出	缓冲区溢出
		整型溢出
		……
	操纵	路径操纵
		配置操纵
		……
	访问控制	密码破解
		锁定不足
		竞争条件
		……
	泄露	资源泄露
		内存泄露
		信息泄露
		……
	文件系统	文件上传
		文件包含
		……
	……	

3)缺陷的表现形式。即缺陷在源代码中表现出的形态，也可以认为是问题代码的呈现状态，这些问题代码在软件运行或测试过程中可能会导致安全问题的出现，当然其中有些是与特定的语言相关的。已经归纳出了预处理、声明和初始化、表达式、整数、浮点数、数组、字符串、内存管理、输入输出、面向对象、环境、并发性、异常处理等 13 类 157 个子类的缺陷表现形式，如表 3 所列。

表 3　缺陷的表现形式

缺陷的表现形式(13 类 157 个子类)	字符串操作	未正确判断字符串长度
		格式化字符串
		字符串迭代
		……
	信号量	单独成员域
		信号处理
		……
	表达式	表达式恒真
		表达式恒假
		……
	异常处理	空 Catch 块
		未处理异常
		过于泛化抛出异常
		……
	数学运算	数学运算符混乱
		类型混用
		除数为零
		……
	常量	常量定义不合理
		常量越界
	……	……

3.2　三维树模型缺陷分类法

在对源代码缺陷分类时，综合考虑上述 3 种属性，即对于某一缺陷，从产生原因、造成的结果和表现形式 3 个维度综合考虑其类别的归属。从表 1－表 3 中可以看出，无论是缺陷产生的原因、造成的结果，还是表现形式，都满足多层次包含关系，这种关系恰好对应于树结构。因此，分别使用一棵树来表示 3 种分类属性，每棵树的叶子节点所组成的集合即对应于属性的最小分类集，如图 2 所示。

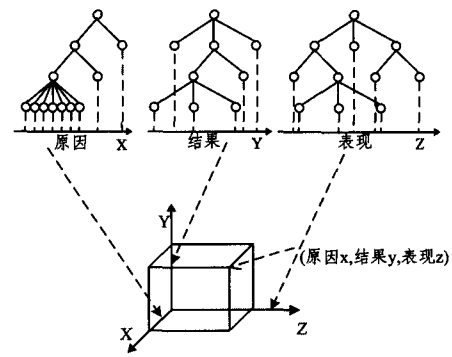

图 2　三维树模型缺陷分类法

定义 1(三维树分类法)　设树 $Tree_{rc}$，$Tree_{rt}$，$Tree_{rf}$ 分别表示源代码缺陷产生的原因(cause)、造成的结果(result)及其表现形式(form of expression)，$child(n)$ 表示树中节点 n 的叶子节点集合，且 $Rc=\{rc\mid(rc\in Tree_{rc})\wedge(child(rc)==\text{Null})\}$，$Rt=\{rt\mid(rt\in Tree_{rt})\wedge(child(rt)==\text{Null})\}$，$Rf=\{rf\mid(rf\in Tree_{rf})\wedge(child(rf)==\text{Null})\}$，则集合 $\{(a,b,c)\mid(a\in Rc)\wedge(b\in Rt)\wedge(c\in Rf)\}$ 是三维树分类法所得的缺陷类集合，任一 (a,b,c) 即为其中的一类源代码缺陷。

基于三维树模型对缺陷进行分类，能够从多个方面反映某类缺陷的信息，详细描述其性质为缺陷检测规则的构造提供基础；另外，基于树型结构描述缺陷的类别便于扩展，当新的缺陷类型出现时，即可在三维树的相应位置添加新节点来表示其 3 方面的属性，便于修改和扩充。

3.3　基于三维树模型的缺陷命名法

在三维树缺陷分类法中，使用原因、结果、表现形式三者的

名称所组成的三元组对该类缺陷进行命名，此命名方式简单、直观、实用，可以通过名称反映出缺陷3个方面的信息，并且易于对缺陷类别进行归纳统计和修改。例如，对于缺陷样本：

```
#include <stdio.h>
int main(){
  char fixed_buf[10];
  sprintf(fixed_buf,"Very long format string\n");
  return 0;
}
```

所描述的缺陷类型，可以使用三元组（注入缺陷及验证缺失，缓冲区溢出，Sprintf()函数中数组越界）来对其进行命名。

4 实例分析

4.1 对CWE缺陷的分类

对CWE中描述的110余种源代码缺陷使用三维树模型分类法进行分类，得到了146类缺陷，表4列出了其中的12类，表中第2列表示CWE中相应ID号的缺陷。

表4 基于三维树分类法划分出的12类CWE缺陷

序号	CWEID	三维树模型缺陷分类法		
		缺陷产生的原因	缺陷造成的结果	缺陷的表现形式
1	113	注入缺陷及验证	HTTP响应截断	不可信输入形成的HTTP Cookie
2	79	注入缺陷及验证	跨站脚本	
3	563	代码风格与质量	死代码	未使用的变量
4	563	代码风格与质量	不恰当的风格	覆盖自增量
5	242,676	应用程序接口错误	危险函数	Mac OS函数
6	242,676	应用程序接口错误	危险函数	Unix函数
7	404	代码风格与质量	错误释放	
8	404,772	边界处理	未释放的资源	文件可能未关闭
9	103	注入缺陷及验证缺失	Struts错误	错误的validate()方法
10	104	注入缺陷及验证缺失	Struts错误	表达没有继承Validation类
11	105	注入缺陷及验证缺失	Struts错误	表达域没有Validator
12	590	注入缺陷及验证缺失	内存误操作	free()的参数为非malloc()开辟的内存

从表4中可以得出以下结论：

1）新分类法能反映缺陷的详细信息。例如，第1类缺陷是由输入引发的、可以造成HTTP响应截断的缺陷，其在代码中表现为不可信数据形成的HTTP Cookie。另外，第3类和第4类两种不同的缺陷在CWE中描述为同一种缺陷“代码风格与质量”（ID为563），但它们在造成的结果和表现形式等方面均有很大差别，新分类法中对此予以区分。

2）新分类法能够体现平台和语言相关等信息。例如，第5类和第6类缺陷都对应于CWE中的242和676号缺陷，但它们分别针对Mac OS和Unix平台。将Mac OS中出现的242和676号缺陷统一命名为第5类，而将Unix中出现的242和676号缺陷统一命名为第6类，以示区别，这样有助于对缺陷进行更详细的分类。第12类缺陷是C/C++语言中特有的。

3）新分类法中的缺陷表现形式属性增加了缺陷的直观性。第9、10、11类缺陷都是由于注入缺陷及验证缺失引起的，造成的结果均为Struts错误，但它们的表现形式却不尽相同，或是由于validate()函数错误，或是由于未继承Validation类，或是由于没有Validator。因此，在缺陷的新分类法中，增加缺陷的表现形式这一属性不仅十分必要，而且使得缺陷的描述更加直观。

4.2 对Fortify缺陷的分类

Fortify是一款知名的源代码安全缺陷分析工具[19]，能够对19种语言进行分析，其定义的缺陷综合了OWASP、CWE等缺陷列表的内容，包括300余种。对其中的86种缺陷使用三维树模型分类法进行分类，得到了194类，表5中列出了其中的10类。表中的每行均代表一种由三维树模型描述的缺陷，该缺陷为相应的Fortify缺陷的一个细分子类。由此可见，使用三维树模型缺陷分类法得到的缺陷类别比Fortify中的类别更加精确和详细。

表5 基于三维树分类法划分出的10类Fortify缺陷

序号	Fortify缺陷	三维树模型缺陷分类法		
		缺陷产生的原因	缺陷造成的结果	缺陷的表现形式
1	跨站脚本	注入缺陷及验证缺失	跨站脚本	弱验证
2	缓冲区溢出	注入缺陷及验证缺失	缓冲区溢出	格式串(%f/%F)
3	死代码	代码风格与质量	死代码	未使用的变量
4	SQL注入	注入缺陷及验证缺失	SQL注入	Hibernate
5	访问控制	访问控制及密码失效	访问控制	匿名的LDAP绑定
6	竞争条件	资源共享与竞争	竞争条件	文件系统访问
7	内存泄露	代码风格与质量	内存泄露	内存二次分配
8	系统信息泄露	边界处理	系统信息泄露	缺少Catch块
9	J2EE错误配置	配置错误	J2EE错误配置	缺少错误处理
10	违背对象模型	应用程序接口错误	违背对象模型	只定义了一个Equals()或GetHashCode()

结束语 本文首先分析了已有软件漏洞分类法的优缺点。在此基础上，提出了一种基于三维树模型的源代码缺陷分类法。最后，使用该方法对CWE和Fortify中描述的源代码缺陷进行了分类。实例分析结果表明：使用该分类法得到的缺陷类别比CWE和Fortify中的缺陷分类更为精确和详细，有助于建立一套比较完善的源代码缺陷分类体系，对预防和修复软件安全漏洞具有重要意义，同时对于缺陷检测规则的细化也具有实际的指导意义。

基于现有工作，未来将进一步细化缺陷分类属性。本文中的分类法是在9类缺陷产生的原因、30余类缺陷造成的结果及150余类缺陷的表现形式的基础上完成的，某些缺陷在

（下转第107页）

[12] Jiang Zheng-wei, Wu Xi-hong, Yang Pei-an, et al. Cloud Provider Selection Method Based on SecSLA[J]. Computer Engineering, 2013, 39(10): 1-5(in Chinese)
姜政伟，巫锡洪，杨沛安，等. 基于 SecSLA 的云供应商选择方法[J]. 计算机工程，2013，39(10)：1-5

[13] Andrieux A, Czajkowski K, Dan A, et al. Web services agreement specification (WS-Agreement) [EB/OL]. http://www.ogf.org/documents/GFD.107.pdf

[14] Henning R R. Security service level agreements: quantifiable security for the enterprise[C]// Proceedings of the 1999 workshop on New Security Paradigms. ACM, 1999: 54-60

[15] Bernsmed K, Jaatun M G, Meland P H, et al. Security SLAs for federated cloud services[C]// 2011 6th International Conference on Availability, Reliability and Security. IEEE, 2011: 202-209

[16] Ludwig H, Keller A, Dan A, et al. Web service level agreement (WSLA) language specification[R]. IBM, 2003: 815-824

[17] Lawrence A, Djemame K, Wäldrich O, et al. Using Service Level Agreements for Optimising Cloud Infrastructure Services[M]// Cezon M, Wolfsthal Y. Towards a Service-Based Internet. Berlin: Springer, 2011: 38-49

[18] Lin Zhi-ming, Mao Zheng-yuan. Comparison Method of Alternatives Advantage Degree for Multiple Attribute Decision-making [J]. Statistics and Decision, 2015(2): 44-47(in Chinese)
林志明，毛政元. 多属性决策的方案比较优势度法[J]. 统计与决策，2015(2)：44-47

[19] Cloud Security Alliance. Security, Trust and Assurance Registry (STAR)[EB/OL]. https://cloudsecurityalliance.org/star

[20] Chen Ai-zu, Tang Wen, Zhang Dong-li. Research on performance evaluation of system operation[M]. Beijing: Science Press, 2009: 56-60(in Chinese)
陈爱祖，唐雯，张冬丽. 系统运行绩效评价研究[M]. 北京：科学出版社，2009：56-60

[21] Li Xiao-lin, Zhang Li-na. Service Selection Strategies Based on Multi-Attribute Group Decision-Making Considering QoS Preference[J]. Computer Systems & pplications, 2014, 23(12): 249-252(in Chinese)
李小林，张力娜. 考虑 QoS 偏好的多属性群决策服务选择策略[J]. 计算机系统应用，2014，23(12)：249-252

(上接第 79 页)

此体系下无法得到准确的定义，如表 4 中第 2 类和第 7 类缺陷的表现形式项为空白。因此，如何进一步细化分类属性将是下一步研究工作的重点。

参 考 文 献

[1] Mei Hong, Wang Qian-xiang, Zhang lu, et al. Software Analysis: A Road Map[J]. Chinese Journal of Compters, 2009, 32(9): 1697-1710(in Chinese)
梅宏，王千祥，张路，等. 软件分析技术进展[J]. 计算机学报，2009，32(9)：1697-1710

[2] Piessens F. A Taxonomy of Causes of Software Vulnerabilities in Internet Software[C]// Proceedings of the 13th International Symposium on Software Reliability Engineering (ISSR'02). 2002: 47-52

[3] Aslam T. A Taxonomy of Security Faults in the Unix Operating System[R]. Technique Report TR-95-09, Department of Computer Science, Purdue University, West Lafayette, USA, 1995

[4] Jiwnani K, Zelkowitz M. Susceptibility Matrix: A New Aid to Software Auditing[J]. IEEE Security and Privacy, 2004, 2(2): 16-21

[5] Landwehr C E, Bull A R, McDermott J P. A Taxonomy of Computer Program Security Flaws with Examples[J]. ACM Computing Surveys, 1994, 26(3): 211-254

[6] Weber S, Karger P A, Paradkar A. A Software Flaw Taxonomy: Aiming Tools at Security[C]// Proceedings of the 2005 Software Engineering for Secure Systems(SESS'05). 2005: 274-281

[7] Tsipenyuk K, Chess B, McGraw G. Seven Pernicious Kingdoms [J]. A Taxonomy of Software Security Errors. IEEE Security & Privacy, 2005, 3(6): 81-84

[8] Power R. Current and Future Danger: A CSI Primer on Computer Crime & Information Warfare[M]. Computer Security Institute, 1998

[9] Krsul I, Spafford E, Tripunitara M. Computer Vulnerability Analysis[R]. Technique Report TR-47909-1398, Department of Computer Science, Purdue University, West Lafayette, USA, 1998

[10] Wenliang D, Mathur A P. Categorization of Software Errors that Lead to Security Breaches[C]// Proceedings of the 21st National Information Systems Security Conference. 1998: 603-612

[11] Bishop M. A Taxonomy of Unix System and Network Vulnerabilities[R]. Technical Report CSE-95-8, Dept. of Computer Science, University of California at Davis, Davis, 1995

[12] Cohen F B. Information System Attacks: A Preliminary Classification Scheme[J]. Computers and Security, 1997, 16(1): 26-49

[13] Howared J D. An Analysis of Security Incidents on the Internet 1989-1995[R]. Pittsburgh, USA: Carnegie Mellon University, 1997

[14] Killourhy K S, Maxion R A, Tan K M. A Defense-centric Taxonomy Based on Attack Manifestations[C]// 2004 International Conference on Dependable Systems and Networks. IEEE, 2004: 102-111

[15] Hansman S, Hunt R. A Taxonomy of Network and Computer Attack[J]. Computers and Security, 2005, 24(1): 31-43

[16] DeMillo R A, Mathur A P. A Grammar-based Fault Classification Scheme and Its Application to the Classification of the Errors of Tex[R]. Technique Report, Department of Computer Science, Purdue University, West Lafayette, USA, 1995

[17] Bazaz A, Arthur J D. Towards a taxonomy of vulnerabilities [C]// Proceedings of the 40th Annual Hawaii International Conference on System Sciences. IEEE, 2007: 163

[18] CWE[OL]. http://cwe.mitre.org

[19] Fortify Software[OL]. http://www.fortify.com

[20] Huang Ming, Zeng Qing-kai. Research on Classification Attributes of Software Vulnerability [J]. Computer Engineering, 2010, 36(1): 184-186(in Chinese)
黄明，曾庆凯. 软件脆弱性分类属性研究[J]. 计算机工程，2010，36(1)：184-186