公 开

开源软件源代码安全漏洞分析报告

——区块链专题

国家互联网应急中心

实验室

2016 年 10 月

目录

1 概述

随着软件技术飞速发展，开源软件已在全球范围内得到了广泛应用。数据显示，从2012 年起，已有超过 80%的商业软件使用开源软件。开源软件的代码一旦存在安全问题，必将造成广泛、严重的影响。为了解开源软件的安全情况，实验室持续对广泛使用的知名开源软件进行源代码安全漏洞分析，并发布季度安全漏洞分析报告。

当下，区块链技术已经不容置疑的成为科技和金融领域最炙手可热的新技术。区块链技术是一种通过去中心化和去信任的方式集体维护一个可靠数据库（账本）的解决方案。由于其高效、便捷、安全的特性，区块链技术受到了金融业的广泛关注。在今年的华盛顿国际金融年会上，美联储评价区块链“可能代表着多年以来，支付、清算及结算领域内最为显著的发展”。

据悉，全世界各大金融巨头都在纷纷大举进军区块链技术领域。各大股票和商品交易所一直积极地通过试验和投资对区块链进行调研。例如，纳斯达克交易所在去年秋季推出了其区块链项目 Linq。早在去年 12 月 30 日，纳斯达克就完成了基于区块链技术平台的首个证券交易。包括我国在内的全球央行也在紧锣密鼓的开展区块链技术相关的研究和应用试点。今年 1 月份，包括摩根大通、花旗集团、德意志交易所集团等在内的多家世界顶级投行联合向区块链项目投资，旨在用来规划建设更加高效和安全的区块链相关金融产品组合。除此之外，巴克莱银行、瑞士信贷集团等 9 家全球顶级银行已着手为区块链技术在银行业中的使用制定行业标准和协议。预计到 2017 年，全球银行业在区块链技术开发中的投入将超过 10 亿美元，领跑所有企业软件板块的发展速度。

今年 6 月，包括中国代表在内的来自 90 个国家的央行及监管机构代表齐聚华盛顿美联储总部，共同探讨区块链技术的发展和应用。可见，区块链的重要性已获得金融监

管者的认可，全球对区块链的兴趣已经不再局限于某个机构内部，开始了大规模的协同探索。而区块链技术的安全是被关注的重点问题，一旦相关软件存在漏洞，将造成巨大的财产损失。

实验室本季度聚焦区块链领域的知名开源软件。结合漏洞扫描工具和人工审计的结果，形成了本漏洞分析报告。本次检测在代码层面发现高危安全漏洞和安全隐患共 746 个。从结果来看，开源区块链软件存在着不容忽视的严重安全风险。

2 被测开源软件

综合考虑用户数量、受关注程度以及更新频率等情况，实验室选取了 25 款具有代表性的区块链软件。表 1 列出了本次被测的开源区块链软件项目的概况。本次检测的软件涵盖了 C，C++， Java，Python，PHP 等编程语言。这些开源软件项目都是国际、国内知名的，拥有广泛用户的软件项目，其中不乏由知名软件公司开发的软件。由于这些软件多数与资产、货币交易相关，一旦出现漏洞，可能会导致财产损失的严重风险。

表 1 被测开源软件项目概览

项目名称	版本	功能描述	主要编程语言	代码行数	Github star 数[1]
Dogecoin	1.10.0	Dogecoin（狗币）是一种开源的点对点的数字货币	C/C++	75,487	1,151
Ripple	0.32.1	点对点支付网络	Java C/C++	31,547	1,125
Litecoin	0.10.2.2 0	一款点对点的分布式网络货币系统	C/C++	27,600	1,115
Mist Browser	1.1	浏览和使用	JavaScript	2,772	1,080

[1]通常认为，项目的 star 越高，则项目被越多的开发者关注，其影响力也越大。

		Ethereum Wallet Đapp 的工具			
Webthree-umbrella	1.3.0	Ethereum 的 C++实现	C++	100,785	693
Web3.js	0.17.0-alpha	Ethereum 的 JavaScript 接口	JavaScript	16,956	389
Solidity	0.4.3	编程语言 Solidity，类似 JavaScript	C++	44,407	339
Dash	0.12.0.58	一种匿名、实时支付的实验性的数字货币	C/C++	96,145	326
Ethereumj	1.3.0	Ethereum yellowpaper 的 Java 实现	Java	83,583	324
Bitmonero	0.9.4	一种使用 CryptoNote 协议的新的密码货币	C/C++	210,269	304
Thunder	0.6.0	支持链外交易的比特币即时支付平台	Java	23,395	225
Bitshares-0.x	Xts/18	金融智能服务平台（老版本）	C++	157,801	173
Btcrelay	0.1	用于比特币简单支付验证（SPV）的 Ethereum 合约	Python	4,223	153
Bitcoin Block Explorer	1.0	一款 Bitcoin 区块链浏览器	PHP	984	140
Ethereum Wallet Đapp	1.3.0	Ethereum(比特币的竞争对手）钱包	JavaScript HTML	2,302	132
Bitshares-2	2.0.160813	金融智能服务平台	C++	131,585	110
Omni Wallet	0.27.0	Mastercoin 网络钱包	JavaScript Python	4,316	77
Bitshares-2-ui	2.0.160514	Bitshares-2 的图	JavaScript	6,644	49

		形界面前端			
Hlp-candidate	2.0.0	对 Linux 基金会的 Hyperledger 项目的数字资产项目	Java	11,445	48
Multichain-explorer	0.8pre	基于网络的数据区块链资源管理器	Python	15,186	37
Bitcoin-wallet	5.0	一款供安卓设备使用的比特币钱包	Java (Android)	18,424	26
OmniJ	0.3.7	数字资产和货币交易平台 Omni Layer 的 Java 实现	Java	5,216	24
nem.core	0.6.79	nem core 是 NIS 和 NCC 的重要组成部分	Java	36,200	24
BlockCrawler	1.0	提供比特币的区块链的浏览器	PHP	255	24
Colored-coins-explorer	1.5	充分支持 Colored coins 功能的完整 Bitcoin 区块浏览器	Javascript	1,304	5

3 测试内容

3.1 安全漏洞种类

本次测试涵盖各类常见安全漏洞。根据安全漏洞形成的原因、被利用的可能性、造成的危害程度和解决的难度等因素进行综合考虑，可以将常见的安全漏洞分为九类：

1) 输入验证与表示（Input Validation and Representation）

输入验证与表示问题通常是由特殊字符、编码和数字表示所引起的，这类问题的发生是由于对输入的信任所造成的。这些问题包括 :缓冲区溢出、跨站脚本、SQL 注入、命令注入等。

2) API 误用（API Abuse)

API 是调用者与被调用者之间的一个约定，大多数的 API 滥用是由于调用者没有理解约定的目的所造成的。当使用 API 不当时，也会引发安全问题。

3) 安全特性（Security Features）

该类别主要包含认证、访问控制、机密性、密码使用和特权管理等方面的漏洞。

4) 内存管理（Memory Management）

与内存操作相关的一类漏洞的统称，常见漏洞包括内存泄漏、释放后使用、双重释放等。这类漏洞通常会导致系统运行性能下降、程序崩溃等，是 C/C++ 语言中常见的一类漏洞。

5) 时间和状态（Time and State）

分布式计算与时间和状态有关。线程和进程之间的交互及执行任务的时间顺序往往由共享的状态决定，如信号量、变量、文件系统等。与分布式计算相关的漏洞包括竞态条件、阻塞误用等。

6) 错误和异常处理缺陷（Errors）

这类漏洞与错误和异常处理有关，最常见的一种漏洞是没有恰当的处理错误（或者没有处理错误）从而导致程序运行意外终止，另一种漏洞是产生的错误给潜在的攻击者提供了过多信息。

7) 代码质量问题（Code Quality）

低劣的代码质量会导致不可预测的行为。对于攻击者而言，低劣的代码使他们可以以意想不到的方式威胁系统。常见的该类别漏洞包括死代码、空指针解引用、资源泄漏等。

8) 封装和隐藏缺陷（Encapsulation）

合理的封装意味着区分校验过和未经检验的数据，区分不同用户的数据，或区分用户能看到和不能看到的数据等。常见的漏洞包括隐藏域、信息泄漏、跨站请求伪造等。

9) 代码运行环境的缺陷（Environment）

该类漏洞是源代码之外的问题，例如运行环境配置问题、敏感信息管理问题等，它们对产品的安全仍然是至关重要的。

前八类漏洞与源代码中的安全缺陷相关，它们可以成为恶意攻击的目标，一旦被利用会造成信息泄露、权限提升、命令执行等严重后果。最后一类漏洞描述实际代码之外的安全问题，它们容易造成软件的运行异常、数据丢失等严重问题。

3.2 安全漏洞级别

我们将源代码的安全问题分为三种级别：高危（High）、中等（Medium）和低（Low）。衡量级别的标准包括两个维度，置信程度（confidence）和严重程度（severity）。置信程度是指发现的问题是否准确的可能性，比如将每个 strcpy()调用都标记成缓冲区溢出漏洞的可信程度很低。严重程度是指假设测试技术真实可信的情况下检出问题的严重性，比如缓冲区溢出（buffer overflow）通常是比空指针引用（null pointer dereference）更严重的安全问题。将这两个因素综合起来可以准确的为安全问题划分级别，如图 1 所示。

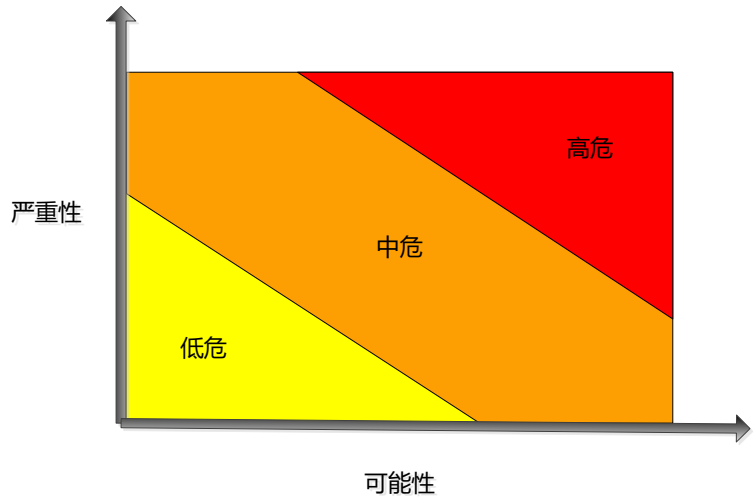

图 1 漏洞级别与严重程度、置信程度的关系

4 开源区块链软件项目的安全漏洞情况

本部分首先展示从被测项目中检出安全漏洞的数量，由此对被测项目的安全性进行大致的评估。然后进一步讨论被测项目中安全漏洞的分布情况，了解项目中出现较多的、容易被忽略的安全问题。

4.1 安全漏洞情况概览

由于高危、中危级别的安全漏洞危害程度较高，更能反映出软件中迫切需要解决的安全问题，本部分展示被测项目中这两种级别漏洞的检出情况，由此对被测项目的安全性进行大致的评估。图 2 展示了被测项目中检出的中高危安全漏洞情况，并对项目按照漏洞数量进行了排序，图中还用红色折线图展示了每千行包含漏洞数[2]。

[2]每千行漏洞数计算方式：漏洞总数/代码行数*1000，精确到小数点后两位

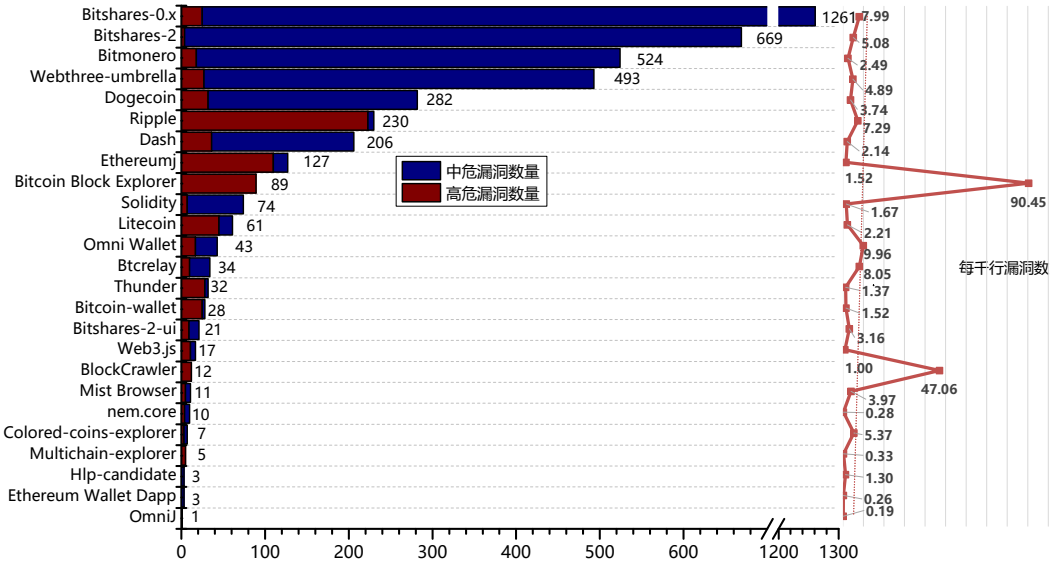

图 2 开源软件项目中高危漏洞情况

从中可以看出，本次选取的区块链软件都存在不同程度的安全问题。本次检测从这些项目中总计发现高危漏洞 746 个，中危漏洞 3497 个。漏洞数量排名靠前的项目处于极易被攻击者利用的状态，实际使用者急需通过安装补丁或者更新版本的方式进行修复和升级。

在所有被测软件中，总体而言安全风险相对较为严重的是区块链支付网络 Ripple，包含高危漏洞 223 个。值得注意的是，该软件很可能是本次检测的区块链相关软件中名气最大、用户最多、使用最广的软件。据悉，截至发稿，该软件所在公司已获得包括 Google、埃森哲等在内共计 1 亿美元的投资，同时一些大型银行已经宣布将加入其支付网络，其中包括渣打、西太银行、澳大利亚国家银行和上海华瑞银行等。考虑到该软件直接处理金融资产，且拥有如此大规模的用户，一旦这些漏洞被黑客利用，将会造成不可估量的损失。

总体安全风险排名第二的是 Ethereumj 项目，该项目是基于区块链的分布式应用平台 Ethereum 的 Java 实现。检测表明，该项目包含高危漏洞 110 个，具有较高的安全风险。

在所有被测软件中，漏洞总数最多的是基于区块链的金融服务软件 BitShares，其最新版本包含中高危漏洞高达 669 个，其中高危漏洞 4 个，中危漏洞 665 个；虽然已经比早期版本(BitShares-0.x 包含高危漏洞 25 个 ,中危漏洞 1236 个)有了明显改善，但仍然存在较大的安全风险。

从漏洞分布密度来看，区块链浏览器 Bitcoin Block Explorer 的漏洞密度最高。其软件规模较小，仅有 984 行代码，但是平均每 11 行代码就存在一个高危漏洞。需要说明的是，该软件 2015 年 9 月后就停止更新和运行了，目前其源代码主要被用作学习和参考。为避免引入不必要的安全风险，关注该项目（Github star 数量为 141）的开发者应知悉其代码中的可能存在的安全隐患。

全部被测软件中，有两款软件不存在高危漏洞，分别是 Ethereum Wallet 和 Hlp-candidate(基于区块链的数字资产)。此外数字资产和货币交易平台 Omni 的 java 实现 OmniJ 只有一个高危漏洞，总体安全性较好。

4.2 高危安全漏洞分布情况

此次测试发现的高危漏洞数量众多。图 3 展示了被测项目中高危漏洞大类的分布情况。数据表明，大多数漏洞为“输入验证与表示”类漏洞，该类漏洞主要是由于对用户输入未做充分验证导致的，一旦攻击者构造恶意输入，可能造成任意命令执行、任意文件读取等严重安全问题。除了包含跨站脚本等传统的“输入验证与表示类”漏洞，导致这种类型漏洞出现较多的一个原因是，部分 Java 语言的区块链软件（如 Ripple）使用

了 JNI 框架，用其他语言（如 C、C++）直接操纵内存等操作系统资源，绕过了 Java 的内存保护机制，使得程序易受到缓冲区溢出等攻击。

“代码质量问题”类漏洞也出现较多，产生的主要原因是开发人员安全意识不足，代码编写不够规范，此类漏洞会导致内存溢出、资源耗尽等安全隐患，严重情况下会导致系统运行异常、甚至系统崩溃。由于区块链软件往往直接运行于金融领域等重要系统中，一旦系统崩溃将带来不能容忍的巨大损失。

“安全特性”类漏洞也占据了一定份额，这类漏洞主要涵盖身份认证、权限管理、密码管理等方面的问题，攻击者可利用该类漏洞实现越权访问，窃取隐私信息等。对于区块链软件来说，加密功能是维护整个开放数据库（账本）安全的核心功能，然而根据此次检测结果，多个软件存在一定数量的“不安全的随机数”问题，这将严重降低软件抵御加密攻击的能力。

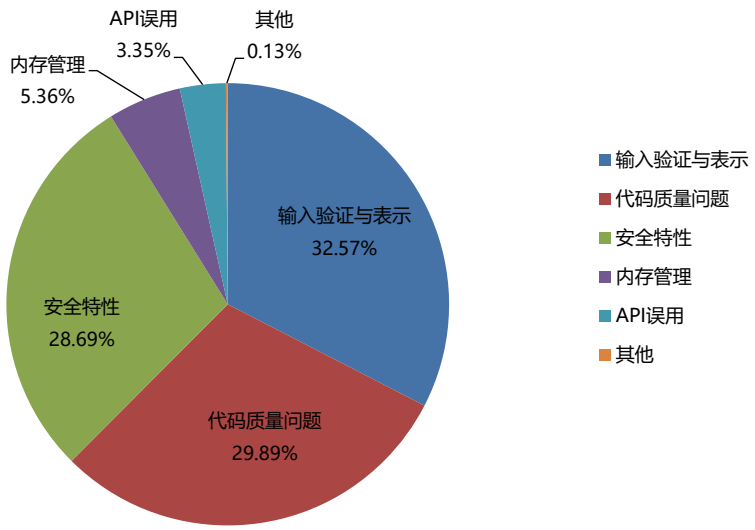

图 3 被测项目中高危安全漏洞的分布情况（按大类划分）

图 4 进一步展示了被测项目中的各种具体的高危级别安全漏洞的分布情况。在被测的 25 个项目中，出现最多的两类漏洞是不安全的 JNI（16.22%，121 个）和不安全的随机数（21.72%，162 个）。下面对这两种漏洞进行简要说明，并给出防范建议。

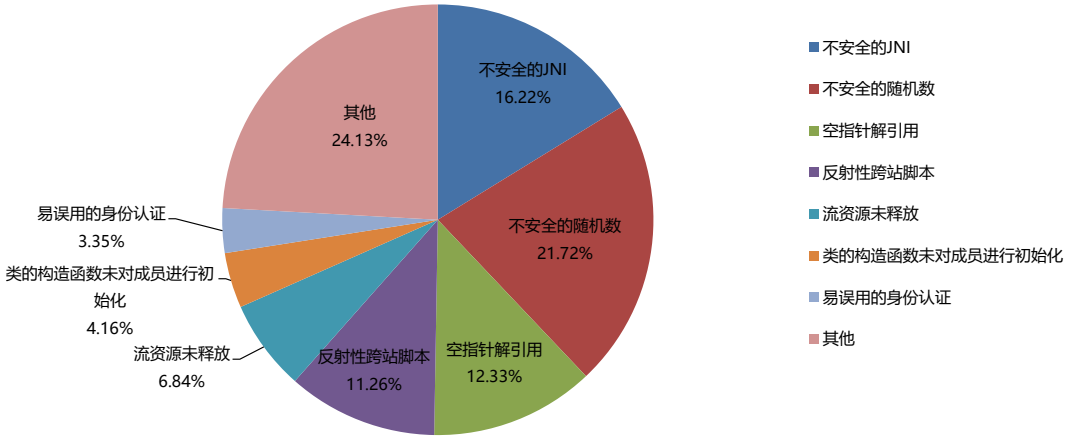

图 4 被测项目中高危安全漏洞的分布情况（按具体漏洞划分）

1）不安全的 JNI（属于输入验证与表示类漏洞）

当 Java 应用程序使用 JNI 调用以其他编程语言编写的代码时，如果应用不当，会导致 Java 应用程序容易受到其他语言的安全漏洞攻击。尽管有 Java 提供的内存保护机制 ,但是该保护机制并不适用于其他语言编写的且通过 Java 本地接口 (JNI) 访问的源代码。

防范：应当认真核查 Java 代码中包含的本地语言所执行的操作，并做好严格的输入验证。

2）不安全的随机数（属于安全特性类漏洞）

在对安全性要求较高的环境中，使用一个能产生可预测数值的函数作为随机数据源，将降低系统抵御加密攻击的能力，导致如易于猜测的密码、可预测的加密密钥、会话劫持攻击和 DNS 欺骗等严重漏洞。

防范：应使用密码学的伪随机数生成器，并使用信息熵最大的信息作为密码学伪随机数生成器的种子。如果信息熵不可用，可以在使用密码学伪随机数生成器的时候改变其种子来降低威胁。

4.3 安全漏洞总体分布情况

上文针对被测项目中的高危漏洞的检出情况对项目的安全状况进行了分析。通常来说，与高危漏洞相比，中低危漏洞在实际运行环境中的危害相对较小，但仍能在一定程度上反映出项目的代码质量、开发人员对代码安全问题的重视程度等。为了更全面的了解被测项目的安全状况，本节进一步展示包括中低危漏洞在内的所有级别安全漏洞的总体分布情况。

图 5 展示了被测项目中安全漏洞大类的分布情况。与高危级别的漏洞分布情况相比，代码质量类问题所占比例大幅提升。查看具体漏洞类型可发现，项目中包含大量的变量未初始化、变量或函数未使用、死代码、缺少对空指针判断等问题，反映了这些软件的代码质量亟待提升。这类问题虽然不会导致直接的严重安全漏洞，但仍然存在着明显的安全隐患，一旦被利用，也可能导致程序崩溃等严重风险。导致代码质量类漏洞较多的一种可能的原因是，此次检测的部分项目是还未正式发布的中间版本，故代码本身还未达到稳定状态，导致部分不完整的“过程代码”的大量留存。

此外，项目还存在相当数量的 API 使用不当问题，例如不安全的使用字符串处理函数、忽略特定 API 的返回值等。未按照约定使用 API，可能导致程序运行发生意想不到的异常问题，从而影响程序逻辑正确性或者系统稳定性。

与高危漏洞分布情况相比，增加了“错误和异常处理”类漏洞。查看具体漏洞类型可以发现，项目中存在一定数量的“空的异常处理代码块”、“捕获过于笼统的异常”等问题，这将导致程序无法正确应对意外情况，使得系统变得脆弱和不稳定，一旦系统运行发生故障，这样草率的异常处理也会使得故障问题的追溯和解决面临困难。事实上，任何的攻击对系统来说都是违背开发者假设的“异常”情况，特别是针对金融系统等对

稳定性和可用性要求极高的行业来说，良好的错误和异常处理更是保障系统安全和稳定的必要条件。

图 6 进一步展示了被测项目中的各种具体的安全漏洞的分布情况。需要说明的是，本次检测结果中出现了 87 种仅出现不超过 10 次的中低危漏洞，如“不恰当的类型转换”、“遗留调试信息“等代码质量、API 使用相关漏洞，为便于数据呈现，在图中统一归到“其他”类型漏洞。在被测的 25 个项目中，出现最多的两种漏洞分别是未使用的局部变量（13.94%，1181 个）、不安全的字符串处理函数（13.20%，1118 个）。下面对这两种漏洞进行简要说明，并给出防范建议。

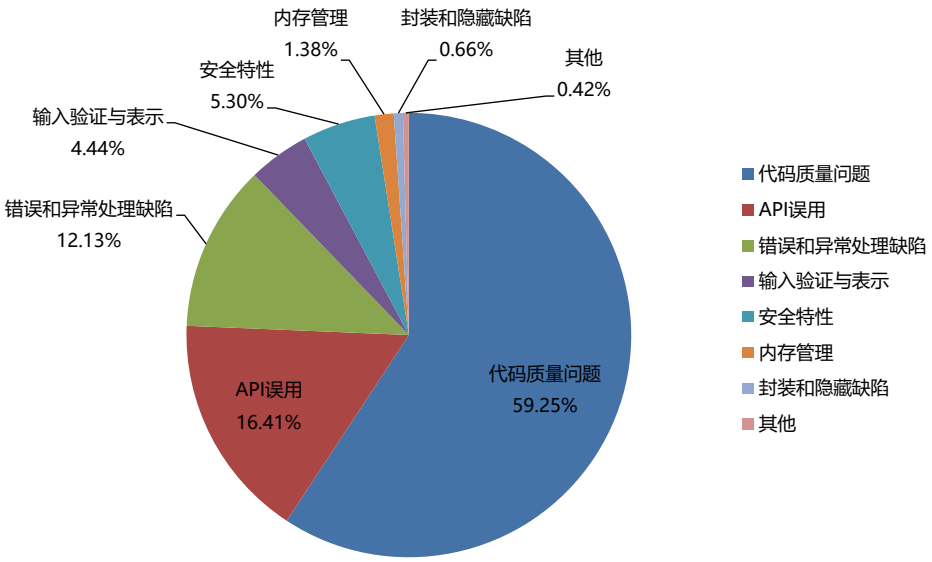

图 5 被测项目中的全部安全漏洞的分布情况（按大类划分）

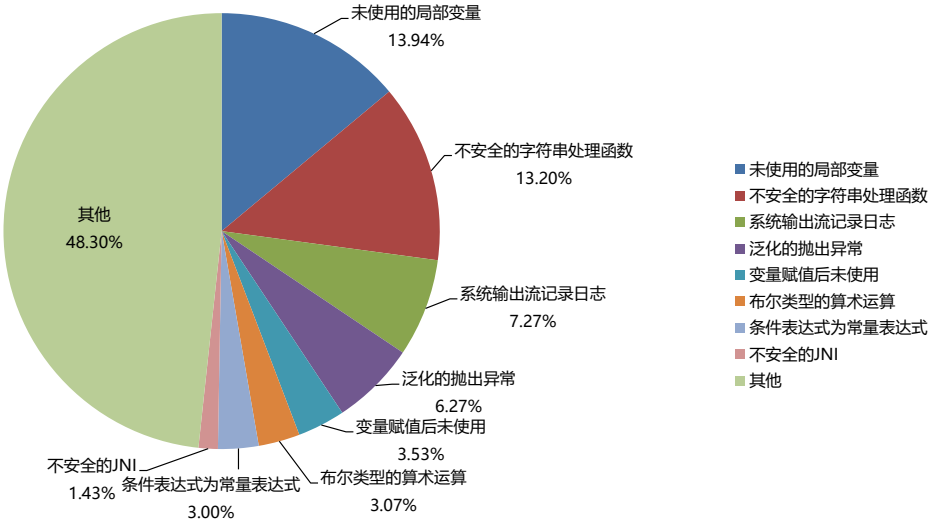

图 6 被测项目中的全部安全漏洞的分布情况（按具体安全漏洞划分）

1）未使用的局部变量（属于代码质量类漏洞）

局部变量在代码中被声明后未被使用，这种问题不会造成直接危害，但通常意味着代码编写过程可能存在逻辑错误，如由于复制粘贴的错误导致使用了错误的变量，或者相关的变量赋值语句被误操作注释掉等。

防范：开发者应检查代码逻辑的完整性和正确性，删除冗余的变量声明。

2）不安全的字符串处理函数（属于 API 误用类漏洞）

有一些字符串处理的标准函数未执行严格的边界检查，存在严重的缓冲区溢出风险，如 C 语言 gets(), strcpy(), strcat(),sprintf(),scanf()等。

防范：应尽量避免使用这些函数 ,使用更安全的替代函数 ,如用 fgets()替代 gets()，同时做好严格的边界检查。

5 关于本报告的说明

一、本报告仅从代码角度进行漏洞分析。本报告中统计的漏洞是指由于代码编写不规范导致的有可能被攻击者利用的安全隐患。在实际系统中，由于软件实际部署环境、安全设备等的限制，部分漏洞可能无法通过渗透测试得到验证。

二、本报告中的漏洞仅适用于表 1 中列出的特定软件版本。当软件版本有任何更新、修改和优化时，本报告不再适用。

三、本报告得到了 360 代码卫士的部分数据支持，在此表示感谢。