

DB32

江 苏 省 地 方 标 准

DB 32/T 4274—2022

工业互联网安全脆弱性分析与检测规范

Industrial Internet Security Vulnerability Analysis and Testing Standards

2022 - 05 - 26 发布

2022 - 06 - 26 实施

江苏省市场监督管理局 发 布

目 次

前言 III

1 范围 1

2 规范性引用文件 1

3 术语、定义和缩略语 1

3.1 术语和定义 1

3.2 缩略语 2

4 总则 3

5 分析与检测流程 3

6 分析 4

6.1 威胁分类与样本库的构建 4

6.2 攻击样本库构建 5

6.3 分析步骤与要求 7

7 检测 13

7.1 弱点脆弱性检测总体要求 13

7.2 环境与封装 13

7.3 API 调用..... 14

7.4 指针安全 15

7.5 初始化与清理环节 16

7.6 错误处理 17

7.7 时间和状态 17

7.8 通用安全特性 19

7.9 数据处理 21

7.10 代码质量 26

7.11 弱点脆弱性检测列表 28

7.12 安全脆弱性检测流程与方法 28

8 服务 29

8.1 服务机构基本能力要求 29

8.2 服务流程 30

附录 A （资料性） 模型的建立..... 31

附录 B （资料性） 工业互联网系统脆弱性检测记录表..... 33

附录 C （资料性） 工业互联网系统脆弱性漏洞访谈信息记录表..... 34

附录 D （资料性） 工业互联网系统脆弱性（漏洞）检测指标及参考权重..... 35

附录 E （规范性） 工业互联网系统脆弱性检测专家打分指标值分类说明..... 36

附录 F （资料性） 判断矩阵表..... 38

附录 G （规范性） 弱点脆弱性检测列表..... 40

附录 H （资料性） 工业互联网系统漏洞严重性等级评定表..... 43

附录 I （资料性） 工业互联网系统脆弱性计算示例..... 44

前 言

本文件按照GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由江苏省信息安全标准化技术委员会提出并归口。

本文件起草单位：南京理工大学、南京工业职业技术大学、五邑大学、东南大学、南京航空航天大学、江苏省工业与信息化厅、江苏省产品质量监督检验研究院、江苏大象信息技术服务有限公司、江苏省中天互联科技有限公司、江苏三台山数据应用研究院有限公司、江苏远恒教育科技有限公司、江苏省信息网络安全协会。

本文件主要起草人：李千目、侯君、金雷、戴晟、武斌、张建航、龙华秋、曹玖新、时宗胜、蒋剑、练智超、韩皓、李冬成、牛博威、孟庆杰、蔡文杰、袁键、邓高见、陈彦文。

工业互联网安全脆弱性分析与检测规范

1 范围

本文件规定了工业互联网系统威胁分类与攻击样本库构建的步骤和一般要求,以及安全脆弱性检测流程与方法、分析与检测服务机构基本能力要求与服务流程要求。

本文件适用于工业互联网系统安全脆弱性分析与检测,脆弱性严重性综合指数计算与等级划分检测。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中,注日期的引用文件,仅该日期对应的版本适用于本文件;不注日期的引用文件,其最新版本(包括所有的修改单)适用于本文件。

- GB/T 25069 信息安全技术 术语
- GB/T 5271.8-2001 信息技术 词汇 第8部分: 安全
- GB/T 20984-2007 信息安全技术 信息安全风险检测规范
- GB/T 25056-2010 信息安全技术 证书认证系统密码及其相关安全技术规范
- GB/T 28452-2012 信息安全技术 应用软件系统通用安全技术要求
- GB/T 302791-2013 信息安全技术 安全漏洞等级划分指南
- GB/T 35273-2017 信息安全技术 个人信息安全规范
- GB/T 30976.1-2014 工业控制系统信息安全

3 术语、定义和缩略语

3.1 术语和定义

下列术语和定义适用于本文件。

3.1.1

工业互联网 Industrial Internet

属于泛互联网的目录分类。使用开放性网络来连接人、数据与机器,激发工业化生产力。

3.1.2

工业互联网系统 Industrial Internet system

面向工业全生命周期,依托工业互联网,在传感网络技术、大数据技术、云计算技术、边缘计算技术、自动化技术、人工智能等技术基础上,通过智能化的感知、人机交互、决策和执行技术,实现设计过程、制造过程和制造装备智能化,是信息技术、智能技术与装备制造技术的深度融合与集成。

3.1.3

漏洞 Vulnerability

计算机信息系统在需求、设计、实现、配置、运行等过程中，有意或无意产生的缺陷。这些缺陷以不同形式存在于计算机信息系统的各个层次和环节之中，一旦被恶意主体所利用，就会对计算机信息系统的安全造成损害，从而影响计算机信息系统的正常运行。保持信息的保密性、完整性、可用性；另外也可包括诸如真实性、可核查性、不可否认性和可靠性等。

[来源：GB/T 30279-2013]

3.1.4

工业控制协议 Industrial control protocol

工业控制系统网络中进行数据交换而建立的规则、标准或约定，规范了设备之间的通信行为与相关的接口标准。

[来源：IEEE802.11]

3.1.5

工业互联网系统安全脆弱性静态分析 Static analysis of industrial Internet system security vulnerability

在不运行工控协议实现程序的前提下，分析软件程序中存在的漏洞，通过对工控协议实现程序的词法、语法、语义进行分析，检测软件中存在的弱安全函数调用和缺陷代码片段。

[来源：GB/T 20278-2013 定义3.3]

3.1.6

工业互联网系统安全脆弱性动态分析 Dynamic Analysis of Security Vulnerability of Industrial Internet System

分别从系统侧和网络侧两方面实现工控协议脆弱性分析的技术。在系统侧，针对工控协议数据的动态污点，通过数据流分析和函数摘要方法，跟踪工控协议数据流的传播，挖掘工控协议实现过程中潜在的漏洞；在网络侧，同时向被测协议的客户端（即工控系统主站）和服务器端（即智能终端设备）发送预先设计的协议畸形数据输入，监视测试过程和返回的异常结果，发现工控协议存在的安全漏洞。

[来源：GB/T 30976.1-2014 定义3.3]

3.2 缩略语

下列缩略语适用于本文件。

ICS：工业控制系统（Industrial Control System）

DCS：分布式控制系统/集散控制系统（Distributed Control System）

PCS：过程控制系统（Process Control System）

PLC：可编程逻辑控制器（Programmable Logic Controller）

RTU：远程终端控制系统（Remote Terminal Unit）

IED：智能电子设备/智能监测单元（Intelligent Electronic Device）

HMI：人机界面（Human Machine Interface）

SIS：生产过程自动化监控和管理系统（Supervisory Information System）

MES：制造执行管理系统（Manufacturing Execution System）
SCADA：数据采集与监视控制系统（Supervisory Control and Data Acquisition）

4 总则

脆弱性检测人员的主要工作职能在于收集和呈现信息，客观地呈现代码安全体系中的问题，不应隐瞒；对代码安全体系等相关内容保守秘密，不对外泄露相关内容。

5 分析与检测流程

工业互联网系统安全脆弱性分析与检测的流程见图1：

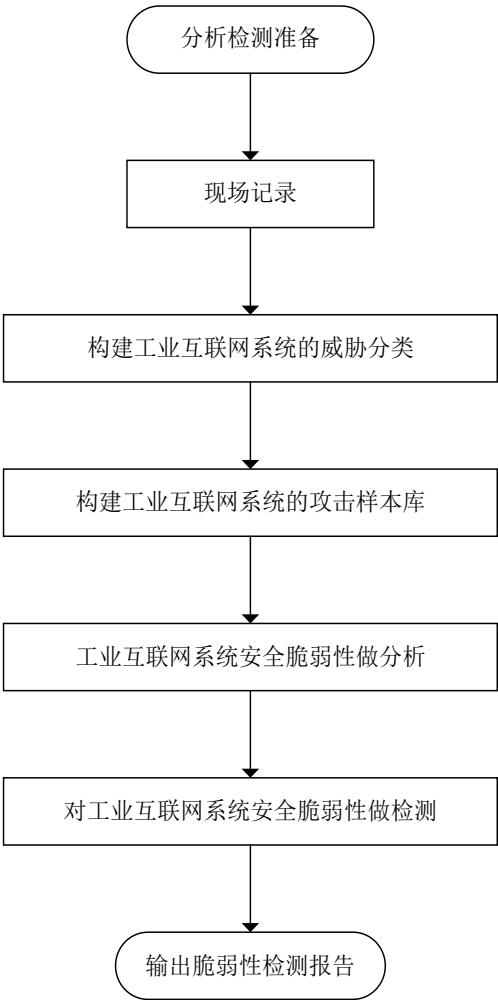


图1 分析与检测流程图

如图1，工业互联网系统安全脆弱性分析与检测的流程为：

- a) 构建工业互联网系统的威胁分类和攻击样本库；
- b) 利用相关方法对工业互联网系统安全脆弱性做分析；

c) 利用相关方法对工业互联网系统安全脆弱性做检测。

6 分析

6.1 威胁分类与样本库的构建

6.1.1 威胁分类原则

表1提供了一种基于表现形式的威胁分类方法。

表1 一种基于表现形式的威胁分类表

种类	描述	威胁子类
软硬件故障	对业务实施或系统运行产生影响的设备硬件故障、通讯链路中断、系统本身或软件缺陷造等问题	设备硬件故障、传输设备故障、存储媒体故障、系统软件故障、应用软件故障、数据库软件故障、开发环境故障
物理环境影响	对信息系统正常运行造成影响的物理环境问题和自然灾害	断电、静电、灰尘、潮湿、温度、鼠蚁虫害、电磁干扰、 洪灾、火灾、地震等
无作为或操作失误	应该执行而没有执行相应的操作，或无意地执行了错误的操作	维护错误、操作失误等
管理不到位	安全管理无法落实或不到位， 从而破坏信息系统正常有序运行	管理制度和策略不完善、管理规程缺失、职责不明确、监督控管机制不健全等
恶意代码	故意在计算机系统上执行恶意任务的程序代码	病毒、特洛伊木马、蠕虫、陷门、间谍软件、窃听软件等
越权或滥用	通过采用一些措施,超越自己的权限访问了本来无权访问的资源,或者滥用自己的职权,做出破坏信息系统的行为	非授权访问网络资源、非授权访问系统资源、滥用权限非正常修改系统配置或数据、滥用权限泄露秘密信息等
网络攻击	利用工具和技术通过网络对信息系统进行攻击和入侵	网络探测和信息采集、漏洞探测、嗅探（账户、口令、权限等）、用户身份伪造和欺骗、用户或业务数据的窃取和破坏、系统运行的控制和破坏等
物理攻击	通过物理的接触造成对软件、硬件、数据的破坏	物理接触、物理破坏、盗窃等
泄密	信息泄露给不应了解的他人	内部信息泄露、外部信息泄露等
篡改	非法修改信息,破坏信息的完整性使系统的安全性降低或信息不可用	篡改网络配置信息、篡改系统配置信息、篡改安全配置信息、篡改用户身份信息或业务数据信息等
抵赖	不承认收到的信息和所作的操作和交易	原发抵赖、接收抵赖、第三方抵赖等

6.1.2 威胁等级

对威胁出现的频率进行等级化处理，不同等级分别代表威胁出现的频率的高低。等级数值越大，威胁出现的频率越高。

表2提供了威胁出现频率的一种威胁等级判断方法。在实际的检测中，威胁频率的判断依据应在检测准备阶段根据历史统计或行业判断予以确定，并得到被检测方的认可。

表2 威胁等级表

等级	标识	定义
5	高	出现的频率很高（或 ≥ 1 次/周）；或在大多数情况下几乎不可避免；或可以证实经常发生过
4	较高	出现的频率较高（或 ≥ 1 次/月）；或在大多数情况下很有可能会发生；或可以证实多次发生过
3	中	出现的频率中等（或 > 1 次/半年）；或在某种情况下可能会发生；或被证实曾经发生过
2	较低	出现的频率较小；或一般不太可能发生；或没有被证实发生过
1	低	威胁几乎不可能发生，仅可能在非常罕见和例外的情况下发生

6.1.3 威胁调查

根据组织和信息系统自身的特点，发生的历史安全事件记录，面临威胁分析等方法进行调查：

- a) 运行过一段时间的信息系统，可根据以往发生的安全事件记录，分析信息系统面临的威胁。例如，系统受到病毒攻击频率，系统不可用频率，系统遭遇黑客攻击频率等等；
- b) 在实际环境中，通过检测工具以及各种日志，可分析信息系统面临的威胁；
- c) 对信息系统而言，可参考组织内其他信息系统面临的威胁来分析本系统所面临威胁；对组织而言，可参考其他类似组织或其他组织类似信息系统面临威胁分析本组织和本系统面临威胁；
- d) 第三方组织发布的安全态势方面的数据。

6.2 攻击样本库构建

6.2.1 攻击样本库分类

6.2.1.1 按攻击目标的不同可分为以下四类：

云攻击、网络攻击、协议攻击和智能端攻击。网络攻击者通常使用公共云环境渗透到本地数据中心进行云攻击。而协议攻击是通过消耗网络服务器、路由器、交换机、防火墙、负载均衡等设备的资源达到破坏的目的。智能端攻击使智能（无人）客户端出现异常。

6.2.1.2 按攻击造成的危害分类可以分为以下五类：

- a) 恶意控制：核心部件被入侵者恶意控制，正常的业务逻辑被篡改，从而对工业互联网系统生产和使用造成巨大的破坏；
- b) 系统停运：入侵者往往采用简单粗暴的方式，向平台系统中关键部件发送大量的非正常数据包，使得目标部件失去功能，使得系统的完整性遭到破坏；
- c) 网络中断：不同于系统停运，入侵者虽然无法控制目标设备，也无法使其瘫痪，但是可以通过干扰其网络连接，使它无法和其他设备通信。尤其在系统网络中广泛使用无线连接，为这类攻击手段提供了应用场景；
- d) 数据篡改：系统中传输和存储的信息数据是工业互联网系统实现自动化管控的基础，信息被入侵者篡改；

- e) 非法接入：智能表计、智能家电、分布式能源设备等多种智能终端大量接入。业务终端数量庞大、类型多样，存在信息泄露、非法接入、被控制的风险。

6.2.1.3 攻击按步骤分类可以分为以下四类：

准备阶段、入侵渗透阶段、部署阶段和执行阶段。综合以上三个方面利用星形多维的分类方法将攻击样本分类，如图2所示。

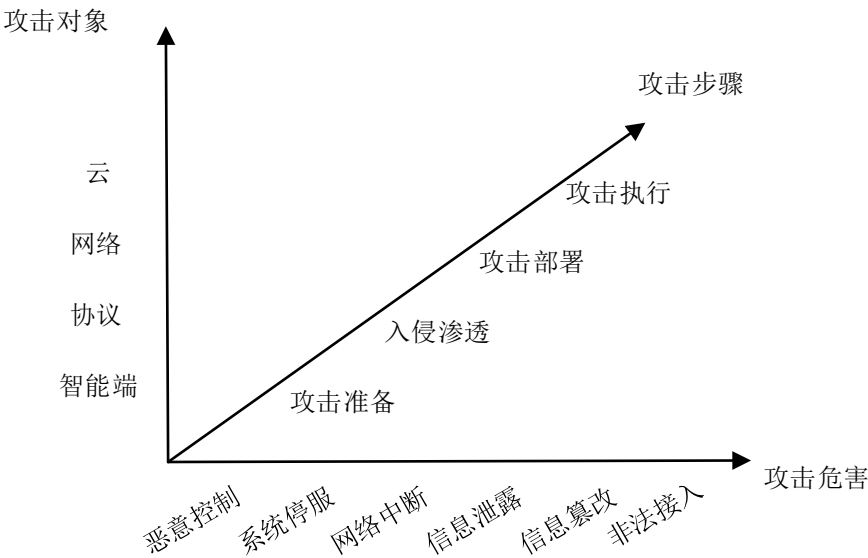


图2 工业互联网系统攻击星形分类方法

6.2.2 攻击样本的构建

基于以上攻击分类标准，服务人员可利用第三方发布的安全数据或利用自身能力挖掘的安全数据，构建符合6.2.3架构的攻击样本库。

6.2.3 攻击样本库的架构

6.2.3.1 攻击样本知识库的构建从六个对象类描述并封装攻击样本，为实现攻击个体以及基于攻击个体的攻击样本的具体化。

6.2.3.2 攻击样本库的体系构架如图 3 所示，其中包括以下六个对象类对应的具体内容：

- a) 具体攻击行为：具体攻击行为描述的是攻击者完成一个原子攻击所进行的操作，可以抽象为对目标建立连接并发送攻击数据包；
- b) 攻击特征：在攻击行为发生时表现出来的系统/数据特征，包括网络流量异常、安全事件报警、网络数据包和主机日志等；
- c) 安全状态：安全状态包含被攻击目标的状态和攻击者的状态，如攻击者当前的起始权限，这是作为攻击行为的前提条件。攻击行为发生后，安全状态改变，作为攻击行为的后果；
- d) 代码实例：代码实例作为具体攻击行为的实现将能够被攻击过程的模拟平台在攻击过程中直接调用；
- e) 安全漏洞：安全漏洞对攻击行为所依附的漏洞宿主进行描述，包括漏洞宿主的软硬件、操作系统和网络应用；

f) 攻击行为间关联关系：为了对攻击过程进行模拟，攻击样本库需要对攻击行为间的前后依赖关系进行描述，以便正确刻画攻击者在某个步骤所能够选择的攻击手段。

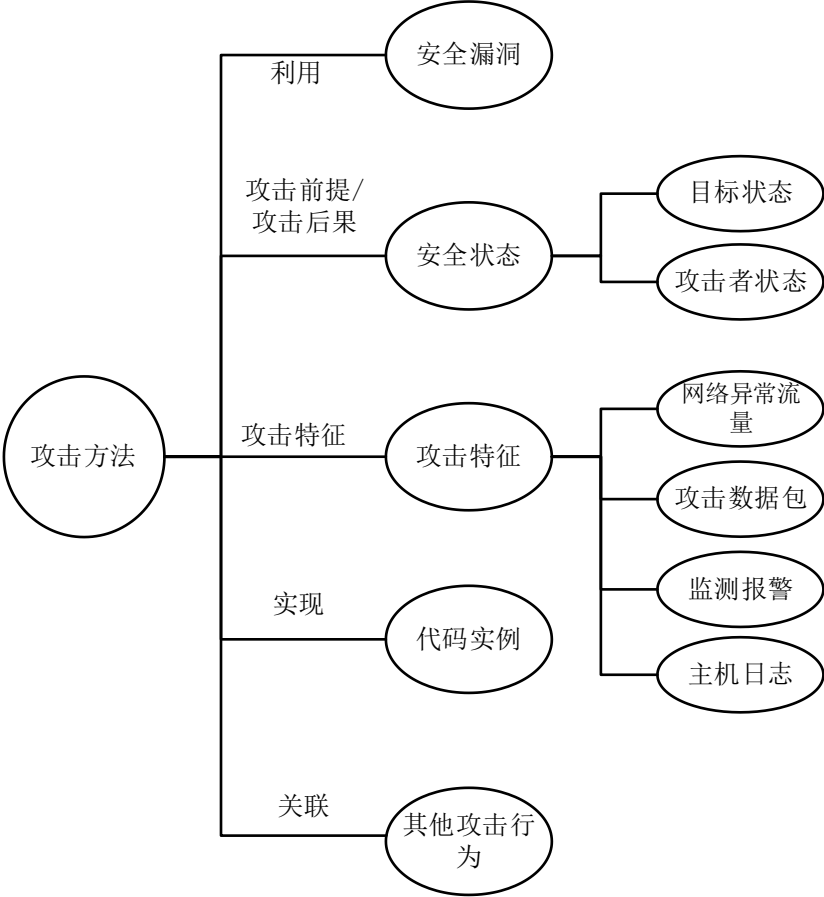


图3 攻击样本库的体系构架

6.3 分析步骤与要求

6.3.1 工业互联网系统安全脆弱性静态分析

6.3.1.1 二进制文件逆向与中间语言表示

6.3.1.1.1 通过逆向分析和中间语言技术，将工控协议实现的二进制文件进行分析，并将其转化成中间语言表示。以中间表示语言为基础，实现模拟执行数据流分析与缺陷的智能识别，进而挖掘漏洞。工控协议脆弱性静态分析流程如图 4 所示。

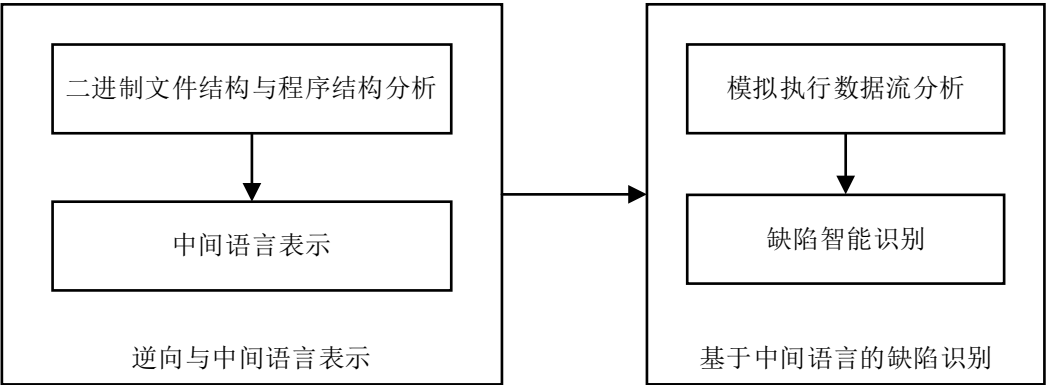


图4 工控协议脆弱性静态分析流程

6.3.1.1.2 二进制文件结构与程序结构分析

根据已知待测二进制文件结构,分析待测二进制代码的文件信息,包括文件总体信息、重定位信息、符号表、调试信息、代码和数据等。

以Mach-O文件格式为例。Mach-O文件格式是 OS X 与 iOS 系统上的可执行文件格式,类似于 windows 的 PE 文件 与 Linux (其他 Unix like) 的 ELF 文件。主要由以下三部分组成:

- 1. Header: 保存了Mach-O的一些基本信息,包括了平台、文件类型、LoadCommands的个数等等。
- 2. LoadCommands: 这一段紧跟Header,加载Mach-O文件时会使用这里的数据来确定内存的分布。
- 3. Data: 这里包含了具体的代码、数据等等。

图5为某一Mach-O文件二进制形式。

pFile	Data LO	Data HI
00000000	<u>CA FE BA BE</u> 00 00 00 02	01 00 00 0C 00 00 00 00
00000010	00 00 10 00 00 27 F0 00	00 00 00 0C 00 00 00 0C
00000020	00 00 00 0B 00 28 00 00	00 23 30 00 00 00 00 0C
00000030	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000050	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000060	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000070	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000080	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000090	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000A0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000B0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000C0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000D0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000E0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000F0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000100	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000110	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000120	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000130	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

图5 某 Mach-O 文件的二进制组成形式

第一行Data LO前四个字节0xCAFEBAFE表示字节的大小端顺序（0xCAFEBAFE为大端，0xBEBAFECA为小端）。后四个字节表示该文件支持哪种CPU架构，图中的0x00000002表示支持的CPU版本。

第一行Data HI前四个字节0x0100000C表示CPU的厂商，后四个字节0x00000000表示CPU类型。

第二行Data LO前四个字节0x00100000表示数据偏移量，即数据记录地址开始的位置。后四个字节表示0x00F02700表示数据量的大小。

第三行Data LO前两个字节0x000B表示符号表数据块结构重定向后的偏移地址。后两个字节0x00280000表示符号的起始地址、字符串的偏移量等。

第三行Data HI前四个字节0x00233000表示文件的通用唯一识别码（UUID），是文件能够被正确序列化的必不可少的条件之一。

6.3.1.1.3 中间语言表示

中间语言是汇编级别的中间表示,指令粒度比汇编指令小，一条汇编指令将映射到若干条中间表示指令中。在翻译过程中，汇编指令的具体语义将被细化并拆分为多条适当的指令，并维持着汇编指令和翻译后的指令的关系。指令集主要包含算术运算操作指令、位操作指令、数据传输操作指令、条件操作指令等。中间语言指令种类如表3所示。

表3 中间语言指令种类

操作种类	操作码	含义	示例
算术运算操作	ADD	两个值相加	ADD t0, t1, t2
	SUB	两个值相减	SUB t6, t8, t9
	MUL	两个值无符号相乘	MUL t3, 4, t5
	DIV	两个值无符号相除	DIV 4000, t2, t3
	MOD	两个值无符号相除求模	MOD t8, 8, t4
	BSH	两个值逻辑移位操作	BSH t1, 2, t2
位操作	AND	两个值按位与操作	AND t0, t1, t2
	OR	两个值按位或操作	OR t7, t9, t12
	XOR	两个值按位异或操作	XOR t8, 4, t9
数据传输操作	STR	存一个值到寄存器	STR t1, t2
	LDM	从内存读取	LDM413600, t1
	STM	存值到内存	STM t2, 414280
条件操作	BISZ	将一个值和 0 比较	BISZ t0, t1
	JCC	条件跳转	JCCT1, 401000
其他操作	UNDEF	解除寄存器值定义	UNDEF, t1
	UNKN	未知	UNKN
	NOP	无操作	NOP

当编程语言被编译后,会产生中间语言表示,可以对中间语言进行分析,进而完成安全脆弱性分析。

例如, ADD t0,t1,t2, 意为将t1、t2寄存器内的值相加,放到t0中去。如果t1、t2寄存器的值非法、不可读或寄存器t0不可写,那么会导致程序存在脆弱性,同时若t0、t1、t2寄存器不做读写检验,则同样存在脆弱性。

6.3.1.2 基于中间语言的缺陷智能识别

6.3.1.2.1 模拟执行数据流分析

模拟执行计划以模块函数调用图和函数控制流图等信息为输入,从模块指定的入口函数开始,迭代往下依次分析每个可达的函数。为了使分析达到最好的效率,内部模拟执行过程应采用函数摘要的策略实现。缺陷查找分析应在模拟执行的前提下进行,模拟执行数据流分析流程图如图6所示。

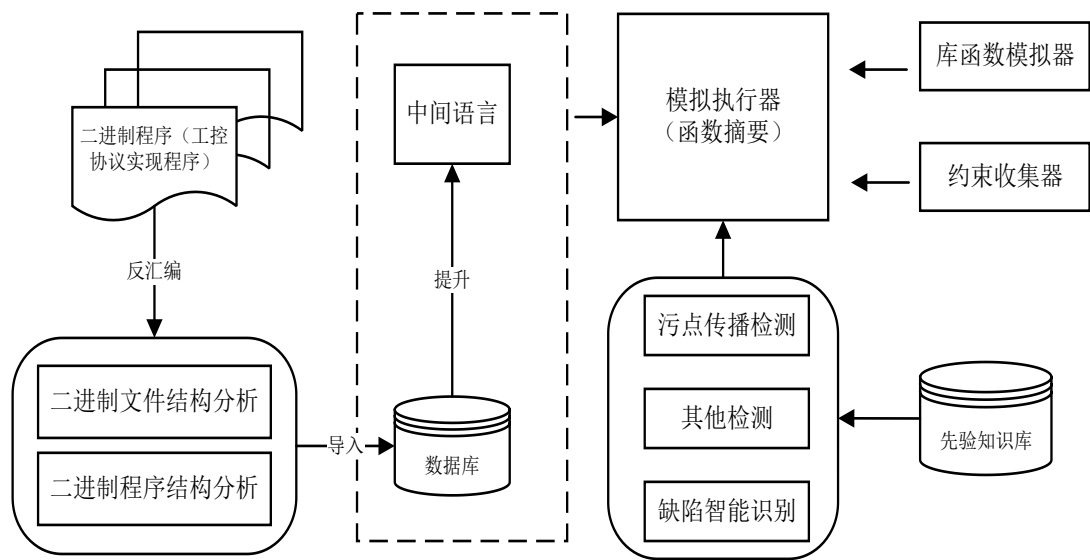


图6 模拟执行数据流分析流程图

6.3.1.2.2 缺陷智能识别

缺陷识别检测是静态脆弱性分析的目的，缺陷检测方法主要基于程序模拟执行获得的模块函数摘要信息，污点传播分析是采用的主要分析策略。

缺陷检测方法深度优先遍历以入口函数为根结点的函数调用树，判断叶子结点是否受污染。污染数据流由根结点进入，一直往下传播至叶子结点。在传播过程中，父函数和子函数间通过子函数摘要中的输入条件映射表进行传递，以父函数的输入污染情况和子函数的输入条件映射表计算子函数的输入污染情况。通过不断迭代传递，将污染数据传至叶子结点（危险调用点或危险代码段）。在叶子结点，进一步判定危险点是否受到污染数据的影响，从而确定是否是一个可疑的安全缺陷点缺陷识别查找的缺陷种类有缓冲区溢出、格式化字符串、注入、命令注入、整数溢出及单字节字符串循环拷贝代码模式。缺陷检测判定对危险缺陷点输入抽象值进行检查，若输入中含有污染值，则较大的可能是确定的危险点，定位为高危危险程度；输入中含有未初始值或未知值，危险程度稍微小些，定位为中级或低级危险。若输入值是未污染的，则可判定该调用点是安全的。

6.3.2 工业互联网系统安全脆弱性动态分析

6.3.2.1 基于动态污点的工控协议动态分析

为检测出引起工控系统运行流程异常的污点数据，需要经过指令识别与污点数据标识、污点数据传播路径跟踪、安全性断言这三个步骤。基于动态污点的工控协议动态分析流程如图7所示。

三个步骤的具体释义如下：

指令识别与污点数据标识：在进行污点分析之前，将目标程序所依赖的数据标记为不可信的污点源数据。

污点数据传播路径跟踪：在污点数据运算的过程中，将所有与之相关的变量、函数等设为污染状态。

安全性断言：在程序的某个运行状态下，检查状态系统中的内存变量或寄存器，判断其中的数据是否被污染以及对应的污点数据是哪些。

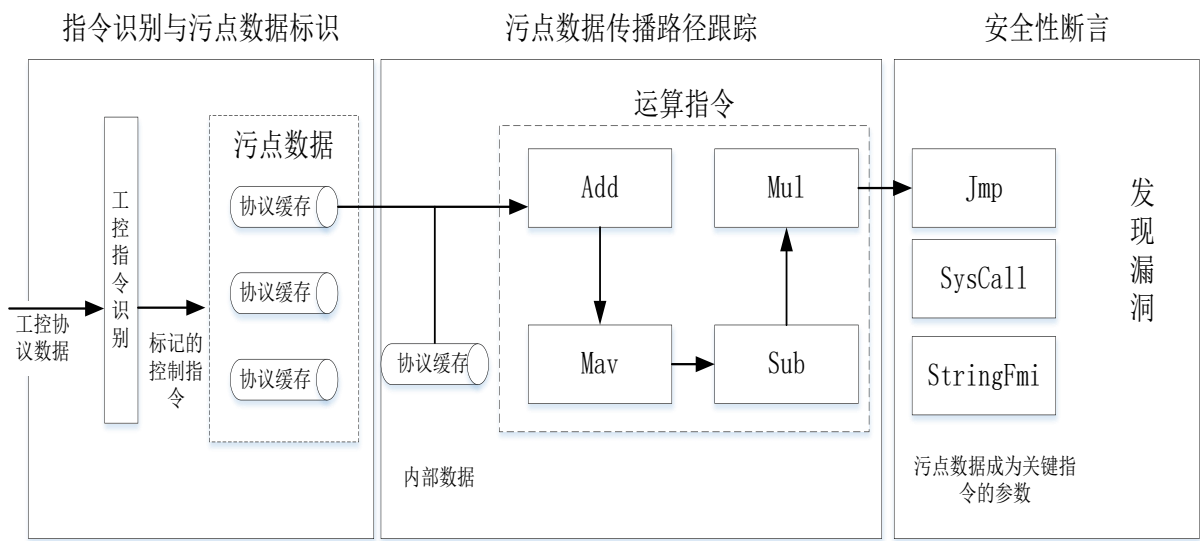


图7 基于动态污点的工控协议动态分析流程

6.3.2.2 基于文法模糊的动态漏洞挖掘技术

基于文法模糊测试的工控协议动态分析流程如图8所示。首先将工控协议抽象为协议结构描述，随后安全协议的结构描述生成模糊测试数据集，模糊测试数据集在测试引擎的调度下通过测试代理向被测系统发送变异的协议数据包，被测系统的状态通过目标监控反馈给测试引擎，以指导后续的调度策略。

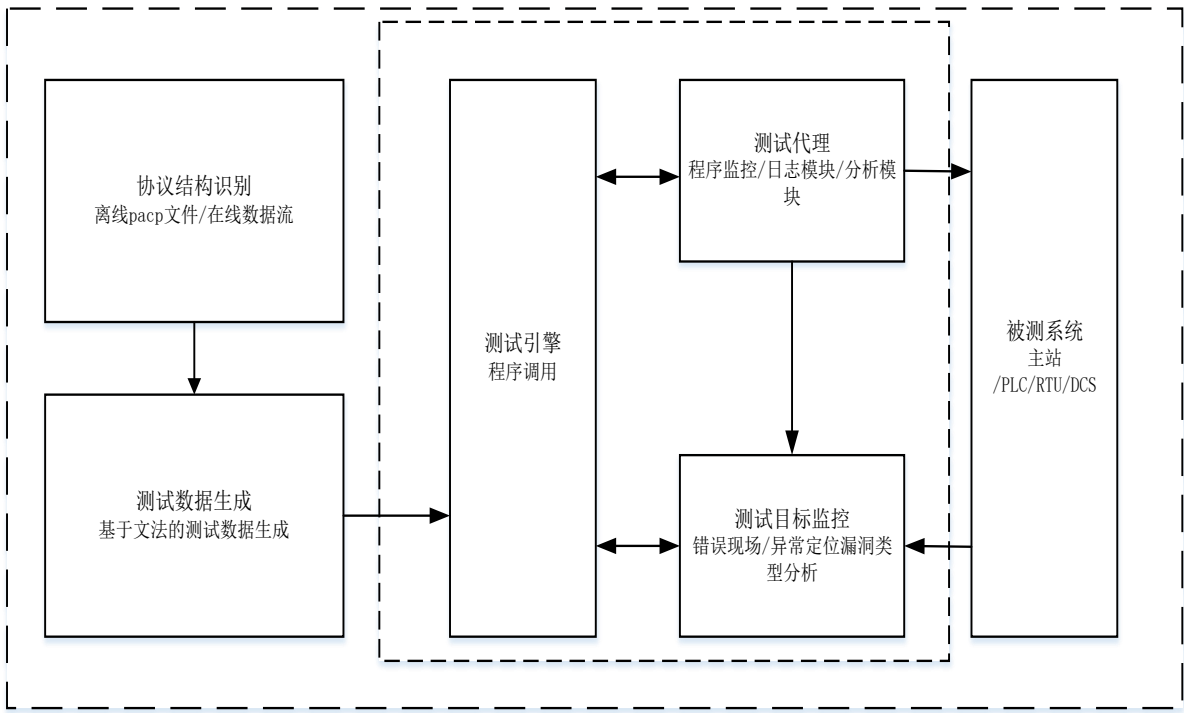


图8 基于文法模糊测试的工控协议动态分析流程

7 检测

7.1 弱点脆弱性检测总体要求

通用弱点脆弱性检测列表是针对常见的桌面应用程序的代码安全体系（以下简称代码）弱点列举的参考列表，特殊应用（如WEB应用、移动应用）的弱点有部分涉及，但不是本章节描述重点。脆弱性检测时可根据被脆弱性检测的具体对象及应用场景对本列表进行调整制定。考虑到语言的多样性，以典型的结构化语言（C）和面向对象语言（Java）为目标进行描述。

本章将从11个方面给出审查工业互联网系统安全的具体条款。其中，环境与封装8条脆弱性检测条款；API调用4条脆弱性检测条款；指针安全7条脆弱性检测条款；初始化与清理环节8条脆弱性检测条款；错误处理1条脆弱性检测条款；时间和状态12条脆弱性检测条款；通用安全特性5类子项，39条脆弱性检测条款；数据处理33条脆弱性检测条款；代码质量17条脆弱性检测条款；总计129条脆弱性检测条款。

7.2 环境与封装

7.2.1 对错误会话暴露数据元素

脆弱性检测指标：代码应避免对错误会话暴露数据元素。

脆弱性检测人员应检查代码在应用的不同会话之间是否会发生信息泄露，尤其是在多线程环境下遗留调试代码。

脆弱性检测指标：代码中不应遗留调试代码。

脆弱性检测人员应检查部署到应用环境的代码安全体系是否含有调试或测试功能的代码，该部分代码是否会造成意外的后门入口。

7.2.2 数据信任边界的违背

脆弱性检测指标：代码应避免将可信和不可信数据组合在同一结构体中，违背信任边界。

脆弱性检测人员应检查代码安全体系是否将来自可信源和非可信源的数据混合在同一数据结构体或同一结构化的消息体中，模糊了二者的边界。

7.2.3 类的错误比较

脆弱性检测指标：在类进行比较时，不应只使用名称比较。

脆弱性检测人员应检查代码中当多个类具有相同的名字时，是否仅通过名字比较类。只通过类名称进行比较可能会导致使用错误的类。如Java语言中应该同时使用 `getClass()` 方法和运算符“`==`”来比较类。

7.2.4 私有数组类型数据域的不安全操作

脆弱性检测指标：公开方法中不应返回私有的数组类型的数据域，或将公共的数据赋值给私有数组类型的数据域。

- a) 脆弱性检测人员应检查代码安全体系声明为公开的方法时，是否返回私有数组的引用，如果结果为肯定，数组的数据会有遭到破坏的安全风险；
- b) 脆弱性检测人员应检查代码安全体系是否将公共数据分配给私有数组，如果结果为肯定，则等于给了该数组公众访问权限，数组的数据会有遭到破坏的安全风险。

7.2.5 包含敏感数据类的安全

脆弱性检测指标：包含敏感数据的类不应可复制和可序列化。

脆弱性检测人员应检查代码安全体系中包含敏感数据的类的相关行为是否安全，具体要求包括但不限于：

- a) 应检查代码安全体系中包含敏感数据的类是否可复制，如 Java 语言中的实现了 Cloneable 接口使类可复制，包含敏感数据的类不应被复制；
- b) 应检查代码安全体系中包含敏感数据的类是否可实现了序列化接口，使类可序列化，包含敏感数据的类不应可序列化。

7.2.6 暴露危险的方法或函数

脆弱性检测指标：不应暴露危险的方法或函数。

脆弱性检测人员应检查代码安全体系中的应用程序编程接口（API）或与外部交互的类似接口是否暴露了危险方法或函数，他们本应受到适当的限制。危险方法或函数暴露的形式有如下几种：

- a) 该方法/函数原本设计为非外部用户使用；
- b) 该方法/函数原本设计为部分用户访问，如限定来自基于 Internet 访问的某网站。

7.2.7 存储不可序列化的对象到磁盘

脆弱性检测指标：不应将不可序列化的对象存储到磁盘。

脆弱性检测人员应检查（如J2EE框架中）代码安全体系是否试图将不可序列化的对象写到磁盘中，将不可序列化的对象存储到磁盘上可能导致序列化失败和应用程序崩溃。

7.3 API 调用

7.3.1 参数指定错误

脆弱性检测指标：函数功能调用应正确指定参数。

脆弱性检测人员应检查函数/方法调用时参数指定是否正确，是否存在如下情况：

- a) 不正确的变量或引用；
- b) 不正确数量的参数；
- c) 参数顺序不正确；
- d) 参数类型不正确；
- e) 错误的值。

以上检查项的任一结果为肯定，则提示存在安全风险。

7.3.2 堆内存释放问题

脆弱性检测指标：应避免在释放堆内存前清理不恰当而导致敏感信息暴露。

脆弱性检测人员应检查代码在释放堆内存前是否采用合适的方式进行信息的清理。如C语言中是否使用realloc()调整存储敏感信息的缓冲区大小，如存在该操作，将存在可能暴露敏感信息的风险。realloc()函数不是从内存中删除，而通常是将旧内存块的内容复制到一个新的、更大的内存块，这可能暴露给攻击者使用“memory dump”或其他方法来进行读取敏感数据的“堆检查”攻击。

7.3.3 忽略函数返回值

脆弱性检测指标：应正确检查函数的返回值，避免忽略函数的返回值。

脆弱性检测人员应检查代码是否对方法或函数调用的返回值作了检查，如未检查函数的返回值，则提示代码存在无法检测非预期状态的风险。

7.3.4 端口多重绑定

脆弱性检测指标：不应在同一端口进行多重绑定。

脆弱性检测人员应检查代码是否有多个套接字绑定到相同端口，从而导致该端口上的服务有被盗或被欺骗的风险。

7.4 指针安全

7.4.1 不兼容的指针类型

脆弱性检测指标：不应使用不兼容类型的指针来访问变量。

脆弱性检测人员应检查代码是否使用不兼容类型的指针来访问变量。通过不可兼容类型的指针修改变量可能会导致不可预测的结果。

7.4.2 利用指针减法确定内存大小

脆弱性检测指标：应避免使用指针的减法来确定内存大小。

脆弱性检测人员应检查代码是否采用一个指针减去另一个指针的方式来确定内存大小。如果两个指针不在相同内存块中，那么使用指针的减法来确定内存大小的计算可能会不正确。

7.4.3 将固定地址赋值给指针

脆弱性检测指标：不应把固定地址赋值给指针。

脆弱性检测人员应检查代码是否将一个NULL或0以外的固定地址赋值给指针。将固定地址赋值给指针会降低代码的可移植性，并为攻击者进行注入代码等攻击提供便利。

7.4.4 试图访问非结构体类型的域

脆弱性检测指标：不应把指向非结构体类型指针强制转换为指向结构类型的指针并访问其字段。

脆弱性检测人员应检查代码是否将指向非结构体类型的指针，强制转换为指向结构类型的指针并访问其字段，如果结果为肯定，则可能存在内存访问错误或数据损坏的风险。

7.4.5 释放一个不在缓冲区起始位置的指针

脆弱性检测指标：不应释放不在缓冲区起始位置的指针。

脆弱性检测人员应检查代码是否释放不在缓冲区起始位置的指针。如果是，则存在应用程序崩溃等风险。

7.4.6 指针偏移越界

脆弱性检测指标：不应使用越界的指针偏移。

脆弱性检测人员应检查代码在使用指针时是否存在偏移越界的现象，因指标偏移越界可能会造成访问缓冲区溢出风险。

7.4.7 无效指针使用

脆弱性检测指标：应避免无效指针的使用。

脆弱性检测人员应检查代码是否存在使用无效指针的现象，因使用无效指针，可能会产生非预期行为。

7.5 初始化与清理环节

7.5.1 资源与变量不安全初始化

脆弱性检测指标：应避免不安全的资源或变量的初始化。

脆弱性检测人员应检查代码安全体系是否对资源或变量进行了安全的初始化，具体要求包括但不限于：

- a) 应检查代码安全体系是否对关键变量进行初始化，未初始化关键变量易导致系统按非预期值执行，如结果为否定，则提示存在安全风险；
- b) 应检查代码安全体系是否采用了不安全或安全性较差的缺省值来初始化内部变量。缺省值通常和产品一起发布，容易被熟悉产品的潜在攻击者获取而带来系统安全风险，如结果为肯定，则提示存在安全风险；
- c) 应检查代码安全体系中关键的内部变量或资源是否采用了可信边界外的外部输入值进行初始化，如结果为肯定，则提示存在安全风险。

7.5.2 引用计数的更新不恰当

脆弱性检测指标：应避免引用计数的更新不恰当。

脆弱性检测人员应检查代码安全体系中管理资源的引用计数是否正确更新，引用计数更新不正确，可能会导致资源在使用阶段就被过早释放，或虽已使用完毕但得不到释放的安全风险。

7.5.3 过期的文件描述符

脆弱性检测指标：不应使用过期的文件描述符。

脆弱性检测人员应检查代码安全体系是否在文件描述符关闭后再次使用或访问它。特定文件或设备的文件描述符被释放后，可被重用，但可能引用了其他的文件或设备。

7.5.4 初始化失败后未退出

脆弱性检测指标：初始化失败后应退出程序。

脆弱性检测人员应检查代码安全体系在初始化失败后能否安全退出。

7.5.5 异常状态的处理

脆弱性检测指标：发生异常时，应恢复对象到先前的状态。

脆弱性检测人员应检查代码安全体系中当异常发生时对象（尤其是关系到系统安全性的对象）能否维持一致的状态。

7.5.6 启用后台线程前主线程未完成类的初始化

脆弱性检测指标：不应在初始化类时启用后台线程。

脆弱性检测人员应检查代码安全体系是否确保主线程完成类的初始化后，再启用后台线程，否则提示系统可能存在类初始化死锁的风险。

7.5.7 发布未完成初始化的对象

脆弱性检测指标：不应发布部分初始化的对象。

脆弱性检测人员应检查代码安全体系在多线程环境中，是否有初始化尚未完成的对象被公开，使得其在本线程初始化完成前就可被其他线程引用。

7.5.8 资源不安全清理

脆弱性检测指标：应避免不安全的资源清理。

脆弱性检测人员应检查代码安全体系中资源清理部分的相关功能，检查项包括但不限于：

- a) 脆弱性检测人员应检查代码安全体系中当特定错误（如不可恢复逻辑错误）发生时，程序是否在以不可预测的状态继续执行，而导致数据有被损坏的风险，在程序终止时应执行正确的清理动作；
- b) 脆弱性检测人员应检查代码安全体系在使用资源后是否恰当地执行临时文件或辅助资源的清理，避免清理环节不完整。

7.6 错误处理

脆弱性检测指标：应恰当进行错误处理。

脆弱性检测人员应检查代码安全体系中错误处理是否安全，具体要求包括但不限于：

- a) 应检查是否对错误进行检查并处理检测到的错误；
- b) 应检查是否采用标准化的、一致的错误处理机制来处理代码中的错误；
- c) 应检查错误发生时，是否提供正确的状态代码或返回值来标示发生的错误；
- d) 应检查是否确保线程池中正在执行的任务失败后给出提示；
- e) 应检查是否对执行文件 I/O 的返回值进行检查；
- f) 应检查是否对函数或操作返回值是否为预期值进行了检查；
- g) 应检查是否返回定制的错误页面给用户来预防敏感信息的泄露。
- h) 如上检查项的任一结果为否定，则提示存在安全风险。

7.7 时间和状态

7.7.1 关键状态数据外部可控

脆弱性检测指标：应避免关键状态数据被外部控制。

脆弱性检测人员应检查代码安全体系中是否将与用户信息或软件自身安全密切相关的状态信息，存储在非授权实体都可以访问的地方，如结果为肯定，则系统可能有关键状态数据能被外部访问或篡改的安全风险。

7.7.2 隐蔽通道

脆弱性检测指标：应避免隐蔽通道。

脆弱性检测人员应检查代码安全体系中是否有隐蔽通道如隐蔽时间通道，如结果为肯定，则提示系统存在信息暴露的安全风险。

注：隐蔽时间通道通过随着时间推移调节某些方面的系统行为来传递信息，接收到信息的程序能够监测系统行为并推断出受保护的信息，从而造成信息暴露。

7.7.3 未加限制的外部可访问锁

脆弱性检测指标：应对外部可访问锁加以限制，不允许被预期范围之外的实体影响。

脆弱性检测人员应检查代码安全体系中的锁是否可被预期范围之外的实体控制或影响，如结果为肯定，则系统存在易受到拒绝服务攻击的安全风险。

7.7.4 会话过期机制缺失

脆弱性检测指标：应制定会话过期机制。

脆弱性检测人员应检查代码安全体系中会话过程是否存在会话过期机制，如结果为否定，则提示代码安全体系存在保护机制被绕过的风险。

7.7.5 将资源暴露给错误范围

脆弱性检测指标：不应将资源暴露给错误的范围。

脆弱性检测人员应检查代码安全体系是否将文件和目录等资源暴露给错误的范围，如果存在，则提示代码安全体系存在信息暴露等风险。例如，一个程序本来意图只是向特定的用户提供某些私密文件，防止攻击者访问这些私密文件。如果文件权限是不安全的，那么特定用户以外的其他用户就有可能访问这些文件。

7.7.6 未经控制的递归

脆弱性检测指标：应避免未经控制的递归。

脆弱性检测人员应检查代码安全体系是否避免未经控制的递归，未控制递归发生的数量可造成预防资源消耗过多的安全风险。

7.7.7 无限循环

脆弱性检测指标：执行迭代或循环应恰当地限制循环执行的次数，应避免无限循环。

脆弱性检测人员应检查代码安全体系中软件执行迭代或循环，是否充分限制循环执行的次数，以避免无限循环的发生或避免攻击者占用过多的资源。

7.7.8 信号错误

脆弱性检测指标：应正确使用信号处理函数。

脆弱性检测人员应检查代码安全体系中信号处理函数的使用，具体要求包括但不限于：

- a) 应检查代码安全体系中如果信号处理函数不可中断，那么信号处理函数在执行时是否已屏蔽该信号，如果结果为否定，则提示存在安全风险；
- b) 应检查代码安全体系中信号处理函数代码序列是否包含非异步安全代码序列，如果结果为肯定，则提示存在安全风险。

7.7.9 共享资源的并发安全问题

脆弱性检测指标：应对共享资源使用正确的并发处理机制。

脆弱性检测人员应检查代码安全体系中共享资源的使用及并发处理的过程，具体要求包括但不限于：

- a) 应检查代码安全体系在多线程环境中对共享数据的访问是否为同步访问，如果结果为否定，则提示存在安全风险；
- b) 应检查代码安全体系中线程间共享的对象是否声明正确的存储持续期，如果结果为否定，则提示存在安全风险；

- c) 应检查代码安全体系中是否在并发上下文中使用不可重入的函数，如果结果为肯定，则提示存在安全风险；
- d) 应检查代码安全体系中是否避免了检查时间与使用时间资源冲突；
- e) 应检查代码安全体系中多个线程中等待彼此释放锁的可执行片段是否避免了死锁情况发生，如果结果为否定，则提示存在安全风险；
- f) 脆弱性检测人员应检查代码对共享资源执行敏感操作时是否检查加锁状态，如果结果为否定，则提示存在安全风险；
- g) 脆弱性检测人员应检查代码是否将敏感数据存储在没有被锁定或被错误锁定的内存中，（将敏感数据存储于加锁不恰当的内存区域，可能会导致该内存通过虚拟内存管理器被写入到在磁盘上的交换文件中，从而使得数据更容易被外部获取），如果结果为肯定，则提示存在安全风险；
- h) 应检查代码安全体系中是否存在互斥体被锁定时删除它们的情况，如果结果为肯定，则提示存在安全风险；
- i) 应检查代码安全体系中是否存在关键资源多重加锁，如果结果为肯定，则提示存在安全风险；
- j) 应检查代码安全体系中是否存在关键资源多重解锁，如果结果为肯定，则提示存在安全风险；
- k) 应检查代码安全体系中是否存在对未加锁的资源进行解锁，如果结果为肯定，则提示存在安全风险；
- l) 应检查代码安全体系中在异常发生时是否释放已经持有的锁，如果结果为否定，则提示存在安全风险。

7.7.10 不安全的临时文件

脆弱性检测指标：应正确使用临时文件。

脆弱性检测人员应检查代码安全体系中临时文件是否安全，预防因临时文件造成的敏感信息泄露，具体要求包括：

- a) 应检查代码安全体系中是否创建或使用不安全的临时文件，如果结果为肯定，则提示存在安全风险；
- b) 应检查代码安全体系中临时文件是否在程序终止前移除，如果结果为否定，则提示存在安全风险；
- c) 应检查代码安全体系中是否在具有不安全权限的目录中创建临时文件，如果结果为肯定，则提示存在安全风险。

7.7.11 符号名称未映射到正确对象

脆弱性检测指标：符号名称应映射到正确对象。

脆弱性检测人员应检查代码安全体系中符号名称是否映射到正确对象，如果结果为否定，则提示存在安全风险。

7.7.12 Web 重定向后执行额外代码

脆弱性检测指标：web应用不应在重定向后执行额外代码。

脆弱性检测人员应检查web代码是否存在重定向后执行额外代码，如果结果为肯定，则提示存在安全风险。

7.8 通用安全特性

7.8.1 权限、特权与访问控制问题

脆弱性检测指标：应确保权限、特权的管理安全以及其他访问控制措施的安全。

脆弱性检测人员应检查代码安全体系中的权限、特权与访问控制功能相关部分，具体要求包括但不限于：

- a) 应检查是否缺失认证机制，如果结果为肯定，则提示存在安全风险；
- b) 应检查是否缺失授权机制，如果结果为肯定，则提示存在安全风险；
- c) 应检查是否存在会话 ID 确定，如果结果为肯定，则提示存在安全风险；
- d) 应检查是否违背最小特权原则，以高于功能所需的特权级别（如根用户或管理员用户）在执行一些操作，如果结果为肯定，则提示存在安全风险；
- e) 应检查放弃特权后，是否检查其放弃是否成功，如果结果为否定，则提示存在安全风险；
- f) 应检查是否创建具有正确访问权限的文件，如果结果为否定，则提示存在安全风险；
- g) 应检查是否避免关键资源的不正确权限授予，如果结果为否定，则提示存在安全风险；
- h) 应检查是否确保正确的行为次序，避免在解析与净化处理之前进行授权，如果结果为否定，则提示存在安全风险；
- i) 应检查是否限制过多认证尝试，如果结果为否定，则提示存在安全风险；
- j) 应检查是否避免使用单一因素认证机制，如果结果为否定，则提示存在安全风险；
- k) 应检查是否可避免攻击者使用欺骗、候选名、候选路径/通道或捕获重放攻击等手段绕过身份认证，如果结果为否定，则提示存在安全风险；
- l) 应检查是否避免不恰当地信任反向 DNS，如果结果为否定，则提示存在安全风险；
- m) 对于客户端/服务器架构的产品，应检查是否避免仅在客户端而非服务器端执行认证，如果结果为否定，则提示存在安全风险；
- n) 应检查是否避免过于严格的账户锁定机制（账户锁定保护机制过于严格且容易被触发，就允许攻击者通过锁定合法用户的账户来拒绝服务合法的系统用户），如果结果为否定，则提示存在安全风险；
- o) 应检查是否在认证机制中使用口令哈希代替口令，如果结果为否定，则提示存在安全风险；
- p) 应检查是否未对信道两端的操作者进行充分的身份认证，或未充分保证信道的完整性，从而允许中间人攻击发生，如果结果为肯定，则提示存在安全风险；
- q) 应检查是否避免通信通道源的验证不当，确保请求来自预期源，如果结果为否定，则提示存在安全风险；
- r) 应检查是否避免通过用户控制密钥绕过授权机制，如果结果为否定，则提示存在安全风险；
- s) 应检查通信信道是否正确指定目的地来预防如下风险：攻击者在目的地伪装成受信任的服务器来窃取数据或引起拒绝服务。如果结果为否定，则提示存在安全风险。

7.8.2 密码安全问题

脆弱性检测指标：密码相关功能应符合国家密码相关标准的要求。

脆弱性检测人员应检查代码安全体系中密码功能是否符合国家密码相关标准的要求，具体要求包括但不限于：

- a) 应检查是否避免在加密中重用密钥；
- b) 应检查是否避免使用已过期的密钥；
- c) 应检查是否避免对敏感数据的明文存储或明文传输；
- d) 应检查是否避免使用已被攻破或存在风险的密码学算法；

- e) 应检查是否选择被业界专家公认比较健壮算法，未自行开发或定制私有加密算法并定期确认加密算法是否已经过期；
 - f) 应检查是否在密码块链接（CBC）加密模式中使用随机初始化向量；
 - g) 应检查 RSA 算法是否结合最优非对称加密填充（OAEP）进行使用。
- 如上检查项的任一结果为否定，则提示存在安全风险。

7.8.3 随机数安全问题

脆弱性检测指标：应确保产生安全的随机数。

脆弱性检测人员应检查代码安全体系是否安全产生随机数，具体脆弱性检测要求包括但不限于：

- a) 应检查是否采用能产生充分信息熵的算法或方案；
- b) 应检查是否避免随机数的空间太小；
- c) 应检查是否避免伪随机数生成器（PRNG）每次都使用相同的种子、可预测的种子（如进程 ID 或系统时间的当前值）或空间太小的种子；
- d) 应检查是否避免使用具有密码学弱点的伪随机数生成器（PRNG）用于加密场景。

如上检查项的任一结果为否定，则提示存在安全风险。

7.8.4 数据真实性验证问题

脆弱性检测指标：应验证数据的起源或真实性，避免接收无效数据。

脆弱性检测人员应检查代码安全体系是否对数据的真实性进行验证，具体脆弱性检测要求包括但不限于：

- a) 应检查是否有数据源或通信源验证错误；
- b) 应检查是否未验证或不正确验证数据的密码学签名；
- c) 应检查是否接受外部混杂在可信数据中的不可信数据；
- d) 应检查是否缺失或进行不恰当完整性检查；
- e) 应检查安全相关的输入是否仅依赖于加密技术而未进行完整性检查；
- f) 应检查是否依赖于文件名或外部提供文件的扩展名而非文件内容进行决策；
- g) 应检查是否信任来自系统事件的信息；
- h) 应检查是否依赖未经验证和完整性检查的 Cookie 进行决策。

如上检查项的任一结果为肯定，则提示存在安全风险。

7.8.5 个人信息保护

脆弱性检测指标：应确保个人信息保护。

脆弱性检测人员应检查代码安全体系是否不恰当地阻止个人隐私信息，在未经明确授权或未经数据相关人的默认同意的情况被访问。不当处理私人信息，可能危及用户隐私，并且可能非法。

7.9 数据处理

7.9.1 非预期数据类型处理不当

脆弱性检测指标：应避免非预期数据类型处理不当。

脆弱性检测人员应检查代码是否正确处理特定元素并非预期类型时的情形，如预期是数字（0-9），但被提供字母（A-Z），代码安全体系是否正确处理。

7.9.2 数据/内存布局问题

脆弱性检测指标：不应依赖数据/内存布局。

脆弱性检测人员应检查代码逻辑是否依赖于对协议数据或内存在底层组织形式的无效假设。当平台或协议版本变动时，数据组织形式可能会发生变化从而带来非预期行为。

7.9.3 绕过净化和验证

脆弱性检测指标：应防止以大小写混合的方式绕过净化和验证。

脆弱性检测人员应检查字符串在查找、替换、比较等操作时，是否存在因大小写问题而被绕过的情况。

7.9.4 在字符串验证前未进行过滤

脆弱性检测指标：不应在过滤字符串之前对字符串进行验证。

脆弱性检测人员应检查对字符串进行验证之前是否存在对该字符串进行过滤，来防止注入类攻击的发生。

7.9.5 条件比较不充分

脆弱性检测指标：执行比较时不应部分比较或不充分的比较。

脆弱性检测人员应检查比较条件是否充分，防止不充分比较造成逻辑绕过风险。

7.9.6 泛型和非泛型原始数据类型

脆弱性检测指标：不应混用具有泛型和非泛型的原始数据类型。

脆弱性检测人员应检查代码是否存在泛型和非泛型之间原始数据类型的混用现象，应避免泛型和非泛型原始数据类型的混用。

7.9.7 字节序使用问题

脆弱性检测指标：应避免字节序使用不正确。

脆弱性检测人员应检查代码在跨平台或网络通信处理输入时是否考虑到字节顺序，避免字节序使用不正确。

7.9.8 结构体长度

脆弱性检测指标：不应将结构体的长度等同于其各个成员长度之和。

脆弱性检测人员应检查代码是否将结构体的长度等同于其各成员长度之和，不应将结构体长度等同于各成员长度之和。结构对象可能存在无名的填充字符而造成结构体长度与各个成员长度之和并不相等。

7.9.9 数值越界回绕错误

脆弱性检测指标：应避免越界回绕错误。

脆弱性检测人员应检查代码是否存在数值赋值超出数值类型范围，应避免赋值越界。

7.9.10 除零问题

脆弱性检测指标：应避免除零错误。

脆弱性检测人员应检查代码是否存在除零操作，应避免除零错误。

7.9.11 边界值检查缺失

脆弱性检测指标：数值范围比较时，不应遗漏边界值检查。

脆弱性检测人员应检查代码在进行数值范围比较时，是否遗漏了最小值、最大值等边界值检查。

7.9.12 敏感信息暴露

脆弱性检测指标：应避免敏感信息暴露。

脆弱性检测人员应检查代码安全体系中是否有敏感信息暴露，重点审查暴露的途径包含但不限于：

- a) 通过发送数据导致的信息暴露；
- b) 通过数据查询导致的信息暴露；
- c) 通过差异性（响应差异性、行为差异性、时间差异性）导致的信息暴露；
- d) 通过错误消息导致的信息暴露；
- e) 敏感数据的不恰当跨边界移除导致信息暴露；
- f) 通过进程信息导致的信息暴露；
- g) 通过调试信息导致的信息暴露；
- h) 在释放前未清除导致信息暴露；
- i) 通过输出流或日志将系统数据暴露到未授权控制的范围；
- j) 通过缓存导致的信息暴露；
- k) 通过日志文件导致的信息暴露；
- l) 通过代码安全体系导致的信息暴露，如测试代码、代码安全体系、注释等；
- m) 敏感信息使用GET请求传递导致信息暴露；
- n) 备份文件导致信息暴露。

7.9.13 信息丢失或遗漏

脆弱性检测指标：应避免信息丢失或遗漏。

脆弱性检测人员应检查代码是否未记录或不恰当记录安全相关信息，应避免信息丢失或遗漏。信息丢失或遗漏导致的常见形式：

- a) 截断与安全有关的信息的显示、记录或处理，掩盖攻击的来源或属性；
- b) 不记录或不显示信息（如日志），而该信息对确定攻击的来源或性质，或确定行动是否安全具有重要意义；
- c) 记录安全相关信息时，使用候选名称而非正式规范的名称导致安全相关信息混淆。

7.9.14 数据结构控制域安全问题

脆弱性检测指标：应避免对数据结构控制域的删除或意外增加。

脆弱性检测人员应检查代码关于数据结构控制域的操作：

- a) 应检查代码是否存在对数据结构控制域的删除而导致系统安全风险。因控制数据被删除，可能会引起严重编程逻辑错误，应避免对数据结构控制域删除。
- b) 应检查代码是否存在对数据结构控制域的意外增加而导致系统安全风险。应避免对数据结构控制域意外增加。

7.9.15 内存缓冲区边界操作

脆弱性检测指标：应避免内存缓冲区边界操作的限制不恰当。

脆弱性检测人员应检查代码在内存缓冲区边界操作时是否存在越界现象，因内存缓冲区访问越界可能会造成缓冲区溢出漏洞。

7.9.16 忽略字符串结尾符

脆弱性检测指标：应保证字符串的存储具有足够的空间容纳字符数据和结尾符。

脆弱性检测人员应检查代码字符串的存储空间是否能容纳下结尾符，字符串不以结尾符结束会造成字符串越界访问。

7.9.17 对环境变量长度做出假设

脆弱性检测指标：不应对环境变量的长度做出假设。

脆弱性检测人员应检查代码在使用环境变量时是否对环境变量的长度做出特定值的假设，因环境变量可由用户进行设置修改，故对环境变量的长度做出假设可能会发生错误。

7.9.18 缓冲区复制造成溢出

脆弱性检测指标：应避免未检查输入数据大小就进行缓冲区复制。

脆弱性检测人员应检查代码在进行缓冲区复制时，是否存在未对输入数据大小进行检查的现象，因未检查输入数据大小，可能会造成缓冲区溢出。

7.9.19 使用错误长度访问缓冲区

脆弱性检测指标：应避免使用错误的长度值访问缓冲区。

脆弱性检测人员应检查代码在访问缓冲区时使用长度值是否正确，因使用错误的长度值来访问缓冲区可能会造成缓冲区溢出风险。

7.9.20 路径遍历

脆弱性检测指标：应避免路径遍历。

脆弱性检测人员应检查代码是否将由外部输入构造的标识文件或目录的路径名，规范化后限制在受限目录，路径遍历会造成非授权访问资源的风险。

路径遍历指未将路径名限制在受限目录。

7.9.21 文件访问解析为链接不恰当

脆弱性检测指标：应避免在文件访问前对链接解析不恰当。

脆弱性检测人员应检查代码基于文件名访问文件时，是否能恰当避免文件名被识别为链接或者快捷方式。链接或快捷方式可能会解析为意想不到的资源。

7.9.22 非可信环境下命令执行

脆弱性检测指标：应避免执行的命令或加载的库文件来自非可信源或在非可信环境中执行。

脆弱性检测人员应检查代码执行的命令或加载的库文件是否来自非可信源，或在非可信环境中执行。如结果为肯定，则应用程序有执行攻击者恶意命令的风险。

7.9.23 循环条件输入导致拒绝服务

脆弱性检测指标：应检查循环条件输入，避免过度循环导致拒绝服务。

脆弱性检测人员应检查代码是否存在循环条件输入，因过度循环条件输入可能会导致拒绝服务风险。

7.9.24 外部控制文件名或路径检查

脆弱性检测指标：应避免文件名或路径的外部可控制。

脆弱性检测人员应检查文件系统操作中的文件名或路径是否被外部控制。若允许用户输入来控制或影响在文件系统操作中所使用的路径或者文件名，则攻击者可能会访问或修改系统文件或对于应用程序至关重要的其他非预期文件。

7.9.25 外部控制格式化字符串问题

脆弱性检测指标：应避免外部控制的格式化字符串。

脆弱性检测人员应检查代码函数接受格式化字符串作为参数，格式化字符串是否来自外部，如果是，则可能有注入类漏洞风险。

7.9.26 对方法或函数参数验证问题

脆弱性检测指标：应对方法或函数的参数进行验证。

脆弱性检测人员应检查代码是否存在对方法或函数的参数进行合法性或安全性校验。

7.9.27 URL 重定向

脆弱性检测指标：不应开放不可信站点的URL重定向。

脆弱性检测人员应检查代码是否存在URL重定向到不可信站点的现象。因重定向到不可信站点，可能会发生跨站伪造请求漏洞风险。

7.9.28 命令行注入

脆弱性检测指标：应正确处理命令中的特殊元素。

脆弱性检测人员应检查代码对利用外部输入来构造命令或部分命令时，是否对其中的特殊元素进行了处理，命令注入通常发生在以下情况中：

- a) 数据从非可信源进入到应用程序中；
- b) 数据是字符串的一部分，该字符串被应用系统当作命令来执行的；
- c) 通过执行这个命令，应用程序为攻击者提供了攻击者不应拥有的权限或能力。

7.9.29 SQL 执行语句注入

脆弱性检测指标：应正确处理SQL命令中的特殊元素。

脆弱性检测人员应检查代码利用用户可控的输入数据构造SQL命令时，是否对外部输入数据中的特殊元素进行处理，如果未处理，那么这些数据有可能被解释为SQL命令而非普通用户的输入数据。攻击者可对此加以利用来修改查询逻辑，从而绕过安全检查或插入可以修改后端数据库的额外语句，可能包括执行操作系统命令。

7.9.30 跨站脚本

脆弱性检测指标：应避免跨站脚本攻击。

脆弱性检测人员应检查代码中当用户提供的数据放到在网页中，被送到浏览器进行显示前，是否进行了验证或过滤。

7.9.31 未恰当处理语法形式不完全符合预期规范的输入

脆弱性检测指标：应恰当处理输入语法形式不完全符合预期规范的输入。

脆弱性检测人员应检查代码中输入功能的部分：

- a) 脆弱性检测人员应检查代码是否正确处理当参数被定义但相关的值缺失情形；
- b) 脆弱性检测人员应检查当超过预期的值被输入时，代码安全体系是否正确处理；
- c) 脆弱性检测人员应检查当输入参数未定义或不支持，代码安全体系是否正确处理；
- d) 脆弱性检测人员应检查代码在函数调用是否对缺失参数的情况进行了处理；
- e) 脆弱性检测人员应检查代码是否处理超过预期数量的参数，应避免未对多余参数做处理；
- f) 脆弱性检测人员应检查代码是否正确处理未定义的参数，应避免未对未定义参数做处理。

7.9.32 对输出日志中特殊字符处理问题

脆弱性检测指标：应对输出日志中的特殊字符进行过滤和验证。

脆弱性检测人员应检查代码是否对输出日志中的特殊字符做过滤和验证。因对特殊字符未做过滤，可能会造成信息泄露。

7.9.33 对 HTTP 头 Web 脚本特殊字符处理问题

脆弱性检测指标：应对HTTP头的Web脚本语法中的特殊字符进行过滤和验证。

脆弱性检测人员应检查代码是否对HTTP头中的Web脚本特殊字符进行过滤处理。因HTTP头中的Web脚本含有特殊字符，可能会导致浏览器执行恶意脚本。

7.10 代码质量

由于有些代码的质量的好坏会影响软件的安全性，本规范将部分代码质量规范纳入标准范畴。

7.10.1 文件描述符穷尽问题

脆弱性检测指标：不应导致文件描述符穷尽。

脆弱性检测人员应检查代码是否存在导致文件描述符穷尽情形。具体检查项包括但不限于：

- a) 是否对打开文件描述符未做关闭处理；
- b) 是否到达关闭阶段之前，失去对文件描述符的所有引用；
- c) 进程完成后是否未关闭文件描述符。

7.10.2 内存未释放

脆弱性检测指标：应及时释放动态分配的内存。

脆弱性检测人员应检查代码是否有动态分配的内存使用完毕后未释放导致内存泄漏的情形。内存泄漏可能会导致资源耗尽从而带来拒绝服务的安全风险。

7.10.3 对网络消息容量的控制

脆弱性检测指标：应避免对网络消息容量的控制不充分。

脆弱性检测人员应检查代码是否控制网络传输流量不超过被允许的值。如果代码安全体系没有机制来跟踪流量传输，系统或应用程序会很容易被滥用于传输大流量(超过了请求值或客户端被允许的值)，从而带来拒绝服务的安全风险。

7.10.4 算法复杂度问题

脆弱性检测指标：应避免算法复杂度攻击。

脆弱性检测人员应检查代码中算法是否在最坏情况下非常低效，复杂度高，会严重降低系统性能。如果是，则攻击者就可以利用精心编制的操作来触发最坏情况的发生，从而引发算法复杂度攻击。

7.10.5 早期放大问题

脆弱性检测指标：应遵守正确的行为次序避免早期放大攻击数据。

脆弱性检测人员应检查代码是否存在允许实体在授权或认证前执行合法但代价高的操作。

7.10.6 子进程访问父进程敏感资源问题

脆弱性检测指标：在调用子进程之前应关闭敏感文件描述符，避免子进程使用这些描述符来执行未经授权的I/O操作。

脆弱性检测人员应检查代码是否存在调用子进程之前有未关闭敏感文件描述符的情形。当一个新进程被创建或执行时，子进程继承任何打开的文件描述符，如不关闭则可能会造成未经授权的访问。

7.10.7 堆空间耗尽问题

脆弱性检测指标：应限制堆空间的消耗，防止堆空间耗尽。

脆弱性检测人员应检查代码是否有导致堆空间耗尽的情形，具体检查项包括但不限于：

- a) 是否存在内存泄漏；
- b) 是否存在死循环；
- c) 不受限制的反序列化；
- d) 创建大量的线程；
- e) 解压一个较大压缩文件。

7.10.8 重复释放资源

脆弱性检测指标：应避免重复释放资源。

脆弱性检测人员应检查代码是否存在重复释放资源的情形。重复释放资源可能会造成系统崩溃。

7.10.9 访问已释放内存

脆弱性检测指标：不应引用或访问已被释放后的内存。

脆弱性检测人员应检查代码是否存在内存被释放再次被访问的情形。内存被释放后再次访问会出现非预期行为。

7.10.10 内存释放指针赋值问题

脆弱性检测指标：应在内存释放立即把指针设置为NULL或指向另一个合法的对象。

脆弱性检测人员应检查代码中内存释放后指针是否未置空或指向另一个合法对象。

7.10.11 内存管理函数成对调用

脆弱性检测指标：应调用匹配的内存管理函数。

脆弱性检测人员应检查代码调用的内存管理函数是否匹配，如malloc/free来分配或删除资源。当内存管理函数不匹配时，可能带来内存损坏或程序崩溃等风险。

7.10.12 条件语句缺失默认情况问题

脆弱性检测指标：switch等条件语句中不应缺失默认情况。

脆弱性检测人员应检查代码switch等条件语句中是否存在缺失默认情况的情形。

7.10.13 无法执行的死代码问题

脆弱性检测指标：不应包含无法执行的死代码

脆弱性检测人员应检查代码是否存在无法执行的死代码。

7.10.14 返回栈上变量地址问题

脆弱性检测指标：不应返回栈上的变量地址

脆弱性检测人员应检查代码在函数中是否存在返回栈变量地址的情形。因栈变量在函数调用结束后就会被释放，再使用该变量地址时会出现意想不到的结果。

7.10.15 可达断言

脆弱性检测指标：不应包含可达断言。

脆弱性检测人员应检查代码在非测试版本中是否包含可达断言。若服务器处理多个并发连接，而断言在其中某个连接中发生，就会导致所有其他连接断掉，从而导致拒绝服务。

7.10.16 实现不一致函数问题

脆弱性检测指标：不应使用具有不一致性实现的函数或字符。

脆弱性检测人员应检查代码是否存在使用了在不同版本具有不一致实现的函数或字符。因使用在不同操作系统或不同版本实现不一致的函数，可能导致代码被移植到不同环境时改变行为。

7.10.17 表达式永真或永假问题

脆弱性检测指标：不应出现表达式永真或永假代码。

脆弱性检测人员应检查代码是否存在表达式逻辑永真或永假代码的情形。

7.11 弱点脆弱性检测列表

详见附录G。

7.12 安全脆弱性检测流程与方法

7.12.1 检测准备

应由检测方、被检测方及必要的第三方（例如系统开发实施单位）组成检测小组。

应明确被检测方工业互联网系统漏洞严重性检测的目标和范围，依据本部分制定检测方案，确定信息系统漏洞严重性的检测清单、检测单位、检测人员清单和检测记录表（格式参见附录B）。

7.12.2 现场记录

组织被检测方人员依据检测方案进行信息系统漏洞信息的获取与搜集，并进行记录（格式参见附录C）。

7.12.3 检测模型的建立

依据现场记录，检测人员应对被检测方信息系统漏洞信息进行确认，建立漏洞严重性检测模型并形成检测过程文档。

以附录A为例，附录A提供了一种“递接层次结构模型”，将其作为漏洞严重性检测模型。其“目标层”为漏洞的严重性；其中间层为该漏洞的可信度、可利用性、修复水平等因素；其方案层为漏洞存在的可能性、技术细节的可信度等因素。这种模型以一种自底向上的方法，将漏洞各方面的因素综合起来，不断向上推进，直至总结为“漏洞的严重性”。

7.12.4 漏洞严重性检测指标项的赋值与确认

依据对被检测方信息系统漏洞信息所建立模型的二级指标进行检测专家赋值并对赋值进行说明，检测组和被检测方的责任人对赋值结果进行确认。工业互联网系统漏洞严重性检测指标及参考权重和检测指标评分准则参见附录D和附录E。

7.12.5 脆弱性（漏洞）状态指数计算

依据建立的检测模型与每类指标的相对重要性，构造判断矩阵表，参见附录F，依次层次分析法与灰色检测方法相结合计算获得每个一级指标的权重分配，然后计算系统漏洞的整体严重性。

7.12.6 漏洞严重性等级确定

依据计算获得的漏洞严重性综合指数，参考工业互联网系统漏洞严重性等级划分表（参见附录H），查表获得信息系统漏洞严重性等级。工业互联网系统漏洞严重性检测计算示例见附录I。

7.12.7 漏洞严重性检测报告

应根据工业互联网系统脆弱性等级和检测过程文档，撰写工业互联网系统脆弱性（漏洞）检测报告。漏洞严重性报告应包括：

- a) 检测方案；
- b) 检测过程；
- c) 检测结果；
- d) 分析与建议。

8 服务

8.1 服务机构基本能力要求

8.1.1 基本资格

服务机构应具备的基本资格包括：

- a) 具有中华人民共和国境内注册的独立法人资格；
- b) 建立人员管理制度和能力考核指标，制定相关培训计划，定期开展培训；

- c) 建立文档管理制度，确保项目文档资料妥善保管；
- d) 建立项目管理制度，有健全的监督检查机制；
- e) 建立保密管理制度，确保客户信息安全可控；
- f) 建立质量管理制度，跟踪服务质量，并能对服务质量进行持续改进。

8.1.2 基本技术能力要求

服务机构应具备的基本技术能力包括：

- a) 具备分析工业互联网系统脆弱性的能力，能够收集、合成系统的脆弱性数据；
- b) 具备分析脆弱性对系统的影响的能力，能够识别脆弱性对所实施的系统的影响，并对发生影响的可能性进行评估；
- c) 具备确定系统脆弱性需求的能力，能够为客户提供策略、目标及需求分析报告；
- d) 具备检测系统脆弱性状况的能力，能对风险变化、事件记录、防护措施进行监视，能识别突发事件和对突发事件进行响应；
- e) 具备检测或证实系统脆弱性的能力，包括检测或证实系统脆弱性的方法和工具。

8.2 服务流程

8.2.1 准备阶段

- a) 服务需求界定：调研客户背景信息，明确客户需求，与客户充分沟通，达成共识并编写需求分析报告；
- b) 服务合同签订：与客户签订服务协议，明确服务范围、目标、时间、内容、金额、质量和输出等。

8.2.2 设计阶段

- a) 服务方案制定：根据客户需求，编制技术方案和实施方案，明确人员、进度、质量、沟通、风险等方面要求。根据项目需求组织客户及相关技术专家对技术方案和实施方案进行论证，确认是否满足要求，编制项目施工手册和作业指导书；
- b) 人员和工具准备：组建服务团队，服务团队应由管理层、相关业务骨干、技术人员等组成。应对服务团队及第三方配合人员进行业务和技能培训。

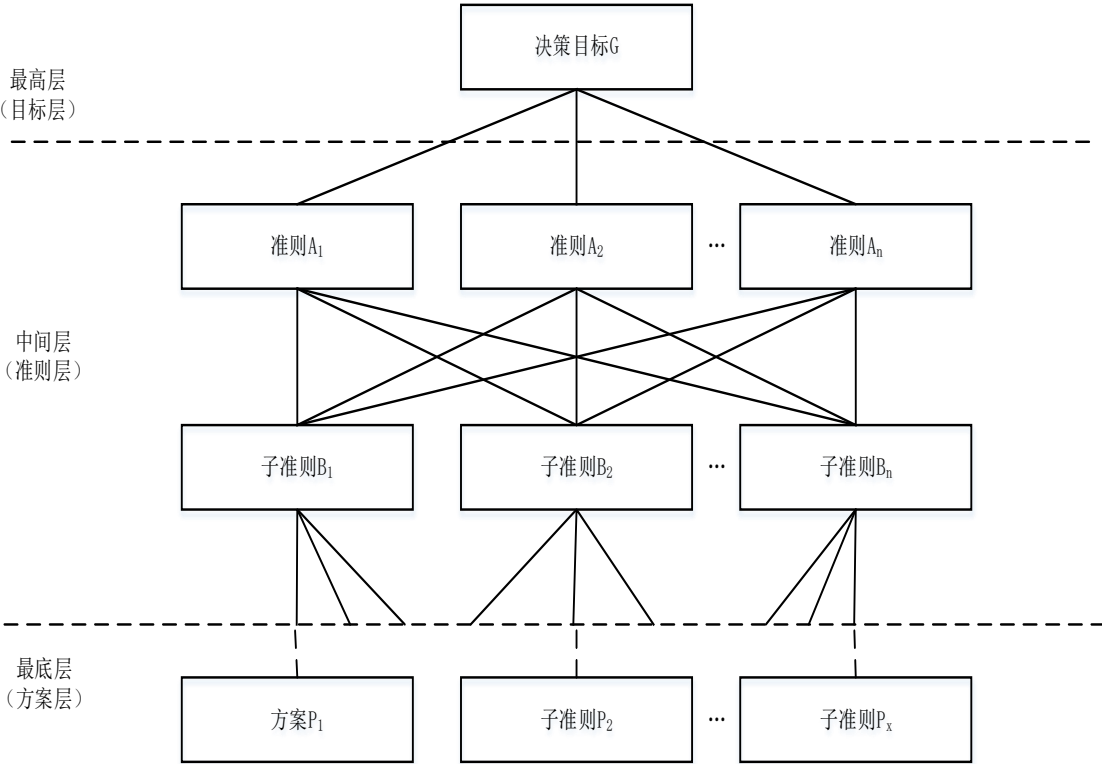
8.2.3 实施阶段

- a) 项目实施人员依照实施方案，按时提交工作日志和记录文档，及时向项目经理汇报项目进度；
- b) 对项目实施监督管理，建立客户满意度调查机制，并对调查结果进行分析；
- c) 根据项目需求和项目范围定期对项目实施情况进行评审，采取适当措施，控制项目风险。

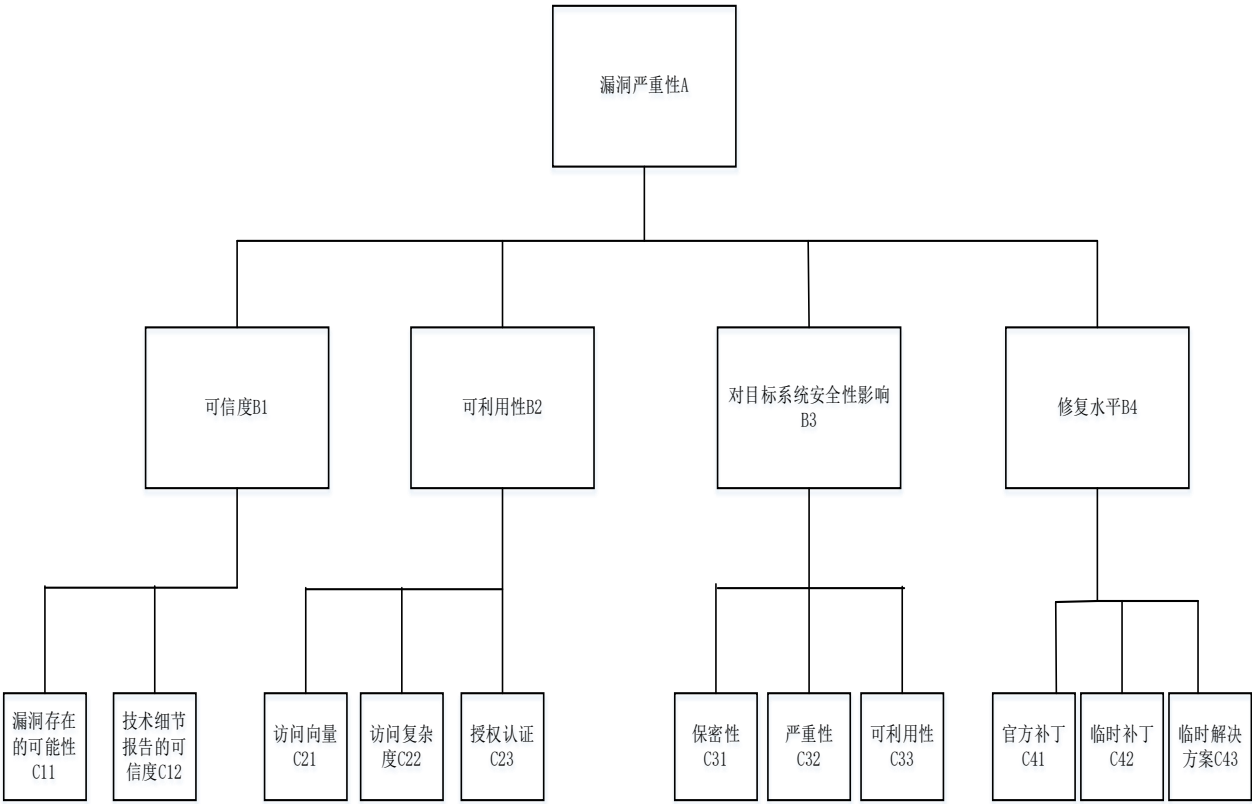
8.2.4 验收阶段

- a) 依照项目需求和项目范围的要求，提出项目初验申请，组织客户和相关方对项目进行初步验收，并提交项目初验报告；
- b) 根据合同约定，配合验收组完成项目终审验收，提交项目终验报告。

附录 A
(资料性)
模型的建立



图A.1 递阶层次结构模型



图A. 2 实际漏洞的结构建立示意图

附 录 B
(资料性)
工业互联网系统脆弱性检测记录表

参见表 B.1

表B.1 工业互联网系统脆弱性检测记录表

被检测系统				检测时间			
被检测单位				服务机构			
检测 方案 概要							
现场 访谈	访谈地点			访谈时间			
	访谈专家	姓名	所属部门		职务	备注	
专家 对漏 洞等 级进 行打 分	二级指标		专家打分				
	漏洞存在的可能性						
	技术细节报告的可信度						
	访问向量						
	访问复杂度						
	授权认证						
	保密性						
	完整性						
	可利用性						
	官方补丁						
	临时补丁						
	临时解决方案						
	漏洞严重性 检测值				漏洞严重性检测结果		
被检测方负责人签字				检测方负责人签字			

附 录 C
(资料性)
工业互联网系统脆弱性漏洞访谈信息记录表

参见表 C.1

表C.1 工业互联网系统脆弱性漏洞访谈信息记录表

被检测系统			访谈时间	
访谈地点			访谈形式	
被检测单位			检测实施单位	
被访谈人			实施访谈人	
访谈主要内容				
访谈主要目的				
漏洞 信息 记录	漏洞名称			
	发布日期			
	影响产品			
	来源			
	描述			
	访问向量			
	访问复杂度			
	授权认证			
	影响类型			
	临时解决方案			
	厂商不定			
被访谈人签字			访谈人签字	

附 录 D
(资料性)

工业互联网系统脆弱性（漏洞）检测指标及参考权重

参见表 D. 1

专家赋值是根据专家本身的经验以及和相应的规则来进行赋值，其中的一些赋值的情况是根据系统本身的状况来进行直接的给定，而一些赋值是根据后文给出的情况来进行判定后进行打分，其中 d_{111} 、 d_{112} 、 d_{123} 、 d_{141} 、 d_{142} ，其他的数值由专家根据相应的表格来进行确定。

表D. 1 工业互联网系统脆弱性（漏洞）检测指标及参考权重

0 级指标	1 级指标	2 级指标	专家赋值	赋值说明
漏洞严重性 A	可信度 B_1	漏洞存在的可能性 C_{11}		
		技术细节报告的可信度 C_{12}		
	可利用 B_2	访问向量 C_{21}		
		访问复杂度 C_{22}		
		授权认证 C_{23}		
	对目标系统安全性影响 B_3	保密性 C_{31}		
		完整性 C_{32}		
		可利用性 C_{33}		
	修复水平 B_4	官方补丁 C_{41}		
		临时补丁 C_{42}		
		临时解决方案 C_{43}		

附 录 E
(规范性)

工业互联网系统脆弱性检测专家打分指标值分类说明

E.1 访问赋值路径说明

访问路径的赋值包括本地、邻接和远程，通常可被远程利用的安全漏洞危害程度高于可被邻接利用的安全漏洞，本地安全漏洞次之。

表E.1 访问赋值路径说明表

赋值	描述
本地	利用该安全漏洞要求攻击者物理接触到受攻击的系统，或者已具有一个本地账号。本地攻击示例：本地权限提升等
邻接	利用该安全漏洞要求攻击者与受攻击系统处于同一个广播域或冲突域中。邻接攻击示例：蓝牙、IEEE802.11 等
远程	不局限与本地和邻接。远程攻击示例：RPC 缓冲区溢出漏洞

E.2 攻击复杂度赋值说明

表E.2 攻击复杂度赋值说明表

赋值	描述
简单	无需借助外部条件，例如自动操作、无需授权即可完成攻击
复杂	需要借助外部条件，例如需要人工参与点击文件，点击按钮或者用户授权

E.3 保密性、完善性和可用性影响赋值说明

影响程度的赋值由安全漏洞对目标的保密性、完整性和可用性三个方面的影响共同导出，每个方面的影响赋值为完全、部分和无。

表E.3 保密性、完善性和可用性影响赋值说明表

赋值	描述
完全	安全漏洞对保密性、完整性和可用性的影响较严重
部分	安全漏洞对保密性、完整性和可用性影响相对较轻
无	安全漏洞对保密性、完整性和可用性无影响

表E.4 访问赋值路径量化表

赋值分类	指标分类量化值	赋值说明
本地	$0 \leq \text{本地} < 3$	
邻接	$3 \leq \text{邻接} < 7$	
远程	$7 \leq \text{远程} < 10$	

表E.5 攻击复杂度量化值表

赋值分类	指标分类量化值	赋值说明
简单	$0 \leq \text{简单} < 5$	
复杂	$5 \leq \text{复杂} < 10$	

表E.6 保密性、完善性和可用性三个方面的量化值

赋值分类	指标分类量化值	赋值说明
无	0	
部分	$1 \leq \text{部分} < 7$	
完全	$7 \leq \text{完全} < 10$	

表E.7 官方补丁、临时补丁、临时解决方案三个方面的量化值

赋值分类	指标分类量化值	赋值说明
没有	0	
有	$7 \leq \text{有} < 10$	

表E.8 漏洞存在的可能性、技术细节报告的可信度的量化值

赋值分类	指标分类量化值	赋值说明
低	$0 \leq \text{低} < 3$	
中	$3 \leq \text{中} < 7$	
高	$7 \leq \text{高} < 10$	

表E.9 授权认证的量化值

赋值分类	指标分类量化值	赋值说明
多次	$0 \leq \text{多次} < 3$	
一次	$3 \leq \text{一次} < 7$	
不需要认证	$7 \leq \text{不需要认证} < 10$	

附 录 F
(资料性)
判断矩阵表

表F.1 第二层中漏洞存在的可能性 C11 和技术细节报告的可信度 C12 相对于可信度 B1 的重要性

C1	名称	漏洞存在的 可能性 C11	技术细节 报告的可 信度 C12
	漏洞存在 的可能性 C11		
	技术细节 报告的可 信度 C12		

表F.2 第二层中访问向量 C21、访问复杂度 C22 和授权认证 C23 相对于可利用 B2 的重要性

C2	名称	访问向量 C21	访问复杂度 C22	授权认证 C23
	访问向量 C21			
	访问复杂度 C22			
	授权认证 C23			

表F.3 第二层中保密性 C31、完整性 C32 和可利用性 C33 相对于对目标系统安全性影响 B3 的重要性

C3	名称	保密性 C31	完整性 C32	可利用 性 C33
	保密性 C31			
	完整性 C32			
	可利用 性 C33			

表F. 4 第二层中官方补丁 C41、临时补丁 C42 和临时解决方案 C43 相对于修复水平 B4 的重要性

C4	名称	官方补丁 C41	临时补丁 C42	临时解决方 案 C43
	官方补 丁 C41			
	临时补 丁 C42			
	临时解 决方案 C43			

表F. 5 表 F. 5 第一层中可信度 B1、可利用 B2、对目标系统安全性影响 B3 和修复水平 B4 相对于漏洞严重性的重要性

B	名称	可信度 B1	可利用 B2	对目标系统安 全性影响 B3	修复水平 B4
	可信度 B1				
	可利用 B2				
	对目标系统安 全性影响 B3				
	修复水平 B4				

附 录 G
(规范性)
弱点脆弱性检测列表

表G.1 弱点脆弱性检测列表

检测	环境与封装	对错误会话暴露数据元素
		遗留调试代码
		数据信任边界的违背
		类的错误比较
		私有数组类型数据域的不安全操作
		包含敏感数据类的安全
		暴露危险的方法或函数
		存储不可序列化的对象到磁盘
	API 调用	参数指定错误
		堆内存释放问题
		忽略函数返回值
		端口多重绑定
	指针安全	不兼容的指针类型
		利用指针减法确定内存大小
		将固定地址赋值给指针
		试图访问非结构体类型的域
		释放一个不在缓冲区起始位置的指针
		指针偏移越界
		无效指针使用
	初始化与清理环节	资源与变量不安全初始化
		引用计数的更新不恰当
		过期的文件描述符
		初始化失败后未退出
		异常状态的处理
		启用后台线程前主线程未完成类的初始化
		发布未完成初始化的对象
		资源不安全清理
	错误处理	
	时间和状态	关键状态数据外部可控
		隐蔽通道
		未加限制的外部可访问锁
		会话过期机制缺失
		将资源暴露给错误范围
		未经控制的递归
		无限循环

		信号错误
		共享资源的并发安全问题
		不安全的临时文件
		符号名称未映射到正确对象
		Web 重定向后执行额外代码
	通用安全特性	权限、特权与访问控制问题
		密码安全问题
		随机数安全问题
		数据真实性验证问题
		个人信息保护
	数据处理	非预期数据类型处理不当
		数据/内存布局问题
		绕过净化和验证
		在字符串验证前未进行过滤
		条件比较不充分
		泛型和非泛型原始数据类型
		字节序使用问题
		结构体长度
		数值越界回绕错误
		除零问题
		边界值检查缺失
		敏感信息暴露
		信息丢失或遗漏
		数据结构控制域安全问题
		内存缓冲区边界操作
		忽略字符串结尾符
		对环境变量长度做出假设
		缓冲区复制造成溢出
		使用错误长度访问缓冲区
		路径遍历
		文件访问解析为链接不恰当
		非可信环境下命令执行
		循环条件输入导致拒绝服务
		外部控制文件名或路径检查
		外部控制格式化字符串问题
		对方法或函数参数验证问题
		URL 重定向
		命令行注入
		SQL 执行语句注入
		跨站脚本

		未恰当处理语法形式不完全符合预期规范的输入
		对输出日志中特殊字符处理问题
		对 HTTP 头 Web 脚本特殊字符处理问题
	代码质量	文件描述符穷尽问题
		内存未释放
		对网络消息容量的控制
		算法复杂度问题
		早期放大问题
		子进程访问父进程敏感资源问题
		堆空间耗尽问题
		重复释放资源
		访问已释放内存
		内存释放指针赋值问题
		内存管理函数成对调用
		条件语句缺失默认情况问题
		无法执行的死代码问题
		返回栈上变量地址问题
		可达断言
		实现不一致函数问题
		表达式永真或永假问题

附 录 H
(资料性)
工业互联网系统漏洞严重性等级评定表

参见表H. 1

表H. 1

漏洞严重性计算量化值	分类	描述
0.0	很低	
2.0	低	
4.0	中	
6.0	较高	
8.0	高	
10.0	很高	

注：计算出的综合量化值与表E. 1的等级量化值进行比较，差的绝对值小的就属于那一个分类。

附 录 I
(资料性)
工业互联网系统脆弱性计算示例

I.1 严重性漏洞检测信息

漏洞名称	Android seccomp 权限提升漏洞
发布日期	2019-05-10
影响产品	Android
来源	国家信息安全漏洞共享平台
描述	Android 一套以 Linux 为基础的开源操作系统。 Android Kernel 组件 seccomp 存在权限提升漏洞。攻击者可利用该漏洞绕过 seccomp，提升权限。
访问向量	远程网络攻击
访问复杂度	低
授权认证	不需要
影响类型	通用型漏洞
临时解决方案	厂商已发布漏洞修复程序，请及时关注更新： https://source.android.com/security/bulletin/2019-05-01
厂商补丁	Android seccomp 权限提升漏洞的补丁

I.2 专家打分获得由各二级指标得分组成的评分样本

0 级指标	1 级指标	2 级指标	专家赋值
漏洞严重性 A	可信度 B ₁	漏洞存在的可能性 C ₁₁	8.0
		技术细节报告的可信度 C ₁₂	8.0
	可利用 B ₂	访问向量 C ₂₁	8.0
		访问复杂度 C ₂₂	9.0
		授权认证 C ₂₃	7.0
	对目标系统安全性影响 B ₃	保密性 C ₃₁	7.5
		完整性 C ₃₂	7.0
		可利用性 C ₃₃	8.0
	修复水平 B ₄	官方补丁 C ₄₁	5.0
		临时补丁 C ₄₂	7.0
		临时解决方案 C ₄₃	9.0

I.3 检测值的计算和级别判定

构造检测体系中每一层的判断矩阵，分别如下：

$$C_1 = \begin{bmatrix} 1 & 4 \\ 1/4 & 1 \end{bmatrix} C_2 = \begin{bmatrix} 1 & 4 \\ 1/4 & 1 \end{bmatrix} C_3 = \begin{bmatrix} 1 & 1/4 & 1/6 \\ 4 & 1 & 1/2 \\ 6 & 2 & 1 \end{bmatrix} C_4 = \begin{bmatrix} 1 & 9 & 5 \\ 1/9 & 1 & 1/4 \\ 1/5 & 4 & 1 \end{bmatrix} B = \begin{bmatrix} 1 & 1/3 & 1/7 & 1/3 \\ 3 & 1 & 1/5 & 2 \\ 7 & 5 & 1 & 5 \\ 3 & 1/2 & 1/5 & 1 \end{bmatrix}$$

1.3.1 一致性验证及确定指标权重

由步骤（1）得到的判断矩阵，计算各层指标权重

$$W_1 = (0.8, 0.2)^T \quad \dots\dots\dots (1)$$

（矩阵 C_1 的特征向量）

$$W_2 = (0.089, 0.3234, 0.5876)^T \quad \dots\dots\dots (2)$$

（矩阵 C_2 的特征向量）

$$W_3 = (0.7439, 0.0633, 0.1939)^T \quad \dots\dots\dots (3)$$

（矩阵 C_3 的特征向量）

$$W_4 = (0.637, 0.1047, 0.2583)^T \quad \dots\dots\dots (4)$$

（矩阵 C_4 的特征向量）

$$W = (0.0614, 0.1811, 0.6294, 0.1281)^T \quad \dots\dots\dots (5)$$

（矩阵 B 的特征向量）

1.3.2 对各判断矩阵进行一致性验证，结果如下

各矩阵的最大特征值：

$$\lambda(C_1)_{\max} \lambda(C_2)_{\max} \lambda(C_3)_{\max} \lambda(C_4)_{\max} \lambda(B)_{\max}$$

各矩阵相对一致性指标:

$$CR(C_1) = CI(C_1) = 0 \leq 0.10$$

$$CR(C_2) = \frac{CI(C_2)}{RI(n=3)} = 0.0079 \leq 0.10$$

$$CR(C_2) = \frac{CI(C_2)}{RI(n=3)} = 0.0079 \leq 0.10$$

$$CR(C_3) = \frac{CI(C_3)}{RI(n=3)} = 0.0614 \leq 0.10$$

$$CR(C_4) = \frac{CI(C_4)}{RI(n=3)} = 0.0332 \leq 0.10$$

$$CR(B) = \frac{CI(B)}{RI(n=4)} = 0.0497 \leq 0.10$$

经验证,各矩阵的相对一致性指标均小于0.10,故各判断矩阵均具有满意的一致性,判断结果合理。

一致性检验RI值

阶数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RI 值	0	0	0.52	0.89	1.12	1.26	1.36	1.41	1.46	0.49	0.52	1.54	1.56	1.58	1.59

假设有一位专家,于是 $M_{ijk}=f_{ij}^k(d_{ij1})$, 请专家打分,获得由各二级指标得分组成的评分样本矩阵 D ,

$$D = \begin{bmatrix} d_{111}, d_{121}, d_{211}, d_{211}, d_{231}, \\ d_{311}, d_{321}, d_{331}, d_{441}, d_{421}, d_{431} \end{bmatrix} = \begin{bmatrix} 8.0, 8.0, 8.0, 9.0, 7.0 \\ 7.5, 7.0, 8.0, 5.0, 7.0, 9.0 \end{bmatrix}$$

把各二级指标的分值代入各灰类的白化权函数,

$$f_j^0 = \begin{cases} 0, x \notin [-2, 2] \\ \frac{x+2}{2}, x \in (-2, 0] \\ \frac{2-x}{2}, x \in (0, 2) \end{cases} f_j^0 = \begin{cases} 0, x \notin [-2, 2] \\ \frac{x+2}{2}, x \in (-2, 0] \\ \frac{2-x}{2}, x \in (0, 2) \end{cases} f_j^1 = \begin{cases} 0, x \notin [0, 4] \\ \frac{x}{2}, x \in (0, 2] \\ \frac{4-x}{2}, x \in (2, 4) \end{cases} f_j^1 = \begin{cases} 0, x \notin [0, 4] \\ \frac{x}{2}, x \in (0, 2] \\ \frac{4-x}{2}, x \in (2, 4) \end{cases}$$

$$f_j^2 = \begin{cases} 0, x \notin [2, 6] \\ \frac{x-2}{2}, x \in (2, 4] \\ \frac{6-x}{2}, x \in (4, 6) \end{cases} f_j^2 = \begin{cases} 0, x \notin [2, 6] \\ \frac{x-2}{2}, x \in (2, 4] \\ \frac{6-x}{2}, x \in (4, 6) \end{cases} f_j^3 = \begin{cases} 0, x \notin [4, 8] \\ \frac{x-4}{2}, x \in (4, 6] \\ \frac{8-x}{2}, x \in (6, 8) \end{cases} f_j^3 = \begin{cases} 0, x \notin [4, 8] \\ \frac{x-4}{2}, x \in (4, 6] \\ \frac{8-x}{2}, x \in (6, 8) \end{cases}$$

$$f_j^4 = \begin{cases} 0, x \notin [6,10] \\ \frac{x-6}{2}, x \in (6,8] \\ \frac{10-x}{2}, x \in (8,10) \end{cases} \quad f_j^4 = \begin{cases} 0, x \notin [6,10] \\ \frac{x-6}{2}, x \in (6,8] \\ \frac{10-x}{2}, x \in (8,10) \end{cases} \quad f_j^5 = \begin{cases} 0, x \notin [8,12] \\ \frac{x-8}{2}, x \in (8,10] \\ \frac{12-x}{2}, x \in (10,12) \end{cases} \quad f_j^5 = \begin{cases} 0, x \notin [8,12] \\ \frac{x-8}{2}, x \in (8,10] \\ \frac{12-x}{2}, x \in (10,12) \end{cases}$$

计算各 2 级指标 C_{ij} 属于各个灰类的灰色评价系数（如表 2.2 所示），

表 1.1 表 2.2 各二级指标属于各个灰类的灰色评价权

白化 指标 权函数	d_{111}	d_{121}	d_{211}	d_{221}	d_{231}	d_{311}	d_{321}	d_{331}	d_{411}	d_{421}	d_{431}
f_0	0	0	0	0	0	0	0	0	0	0	0
f_1	0	0	0	0	0	0	0	0	0	0	0
f_2	0	0	0	0	0	0	0	0	0.5	0	0
f_3	0	0	0	0	0.5	0.25	0.5	0	0.5	0.5	0
f_4	1	1	1	0.5	0.5	0.75	0.5	1	0	0.5	0.5
f_5	0	0	0	0.5	0	0	0	0	0	0	0.5

得到 1 级指标 B_i 的所有 2 级指标 C_{ij} 对于各个评价灰类的灰色评价权矩阵：

$$M_1 = (M_{11}, M_{12})^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$M_2 = (M_{21}, M_{22}, M_{23})^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 \end{bmatrix}$$

$$M_3 = (M_{31}, M_{32}, M_{33})^T = \begin{bmatrix} 0 & 0 & 0 & 0.25 & 0.75 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$M_4 = (M_{41}, M_{42}, M_{43})^T = \begin{bmatrix} 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

(1) 计算 0 级指标 A 下属所有 1 级指标 B_i 对于各评价灰类的灰色评价权矩阵 N

如 $N_1 = W_1 \times M_1$
 $N_1 = W_1 \times M_1$

$$N = [N_1, N_2, N_3, N_4]^T \quad N = [N_1, N_2, N_3, N_4]^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.2938 & 0.5445 & 0.1617 \\ 0 & 0 & 0 & 0.2173 & 0.7827 & 0 \\ 0 & 0 & 0.3185 & 0.3709 & 0.1815 & 0.1291 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.2938 & 0.5445 & 0.1617 \\ 0 & 0 & 0 & 0.2173 & 0.7827 & 0 \\ 0 & 0 & 0.3185 & 0.3709 & 0.1815 & 0.1291 \end{bmatrix}$$

(2) 计算总指标 A 的灰色评价权向量 V

$$V = W^T \times NV = W^T \times N = (0, 0, 0.0408, 0.2375, 0.6759, 0.0458)$$

由

$$\max_{1 \leq i \leq 6} \{v_i\} = v_5 = 0.6759 \quad \max_{1 \leq i \leq 6} \{v_i\} = v_5 = 0.6759$$

可知，该漏洞严重性属于灰类 5（高）的评价权最高，此外该漏洞严重性属于灰类 4（中）的评价权也比较大为 0.3472，仅次于 0.4009，故该漏洞的严重性等级是介于中和高之间且偏向于高。

(3) 计算该漏洞严重性的综合量化值

由

$$S^T = [2.0, 4.0, 6.0, 8.0, 10.0]$$

$$U_V = V \times S^T = 7.4534$$

可得漏洞严重性的综合量化值为 7.4534。