ICS
CCS

T/CMA

中 国 计 量 协 会 团 体 标 准

T/CMA ZK 160—2024

基于可信视频的远程校准系统性能要求

The Performance verification of
Remote Calibration Systems Based on Trusted Videos

2024－XX－XX 发布　　2024－XX－XX 实施

中 国 计 量 协 会　发 布

目　　次

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。

本文件由中国计量协会提出。

本文件由中国计量协会智库工作委员会归口。

本文件起草单位：

本文件主要起草人：

基于可信视频的远程校准系统性能要求

1 范围

本文件规定了主要采用可信视频实现远程校准的系统性能要求，明确远程监测的工作流程，提高远程监测的规范性和操作性，并提高可信度。

本文件适用于主要采用可信视频实现远程校准的系统，采用其它方式实现远程校准的系统性能可以参照本文件的方法进行测评。

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。

ISO/IEC 14496-10 Information technology — Coding of audio-visual objects — Part 10: Advanced video coding

GB/T 25069 信息安全技术 术语

GB/T 32905 信息安全技术 SM3密码杂凑算法

GB/T 35276 信息安全技术 SM2密码算法使用规范

GB/T 35918.2 信息安全技术 SM2椭圆曲线公钥密码算法 第2部分 数字签名算法

GB/T 35918.3 信息安全技术 SM2椭圆曲线公钥密码算法 第3部分 密钥交换协议

GB/T 35918.4 信息安全技术 SM2椭圆曲线公钥密码算法 第4部分 公钥加密算法

GB 35114 公共安全视频监控联网信息安全技术要求

3 术语和定义

GB/T 25069界定的以及下列术语和定义适用于本文件。

3.1 远程检测 Remote measurement

采用异地方式完成的检测活动。

注1：这里的检测包括了校准。校准一般指计量领域的校准活动或CNAS认可的校准活动。

注2：异地指完成检测活动的所有必须要素中至少有一项是在异地。

3.2 可信视频 Trusted video

经过传输以后获得的视频是能被实体机构信任的，包括来源可信任、内容可信任。

注1：传输通过线上或线下完成。

注2：来源可信任指身份真实

注3：内容可信任指没有被篡改

3.3 可信检测 Trust measurement

远程检测是能被实体机构信任的，包括检测活动主体、客体可信任、检测内容可信任。

注1：检测活动主体、客体可信任指主客体身份真实

注2：检测内容可信任指检测的所有要素都没有被篡改

3.4 篡改 Tamper

破坏远程检测活动内容的安全性的行为。

4 远程检测系统的组成要素

远程检测系统是完成原有的线下现场检测活动的线上系统。为在技术上取得与原有的线下现场检测活动等效的校准结果，需具备以下组成要素（见图1）：

a）现场前端层。主要包括现场检测活动的视频子系统，例如传统检测项目多机位多景深现场视频、数字化校准/图像类校准结果多维视频；以及基于IoT架构的探测子系统，例如检测活动直接测量量的探测器、影响量的探测器、环境参数的探测器（温度、湿度等）。

b）可信传输层。主要包括支撑可信检测的时间戳、空间戳、CA、基于国密算法的加密等，并能够在公网进行传输。

c）检测结果后端层。主要功能是给出传输检测的结果，一般采用的标准是标准参考数据（SRD），采用的形式是数字校准证书（DCC）。可以具备实现检测结果到现场前端层的远程计量、在线计量等功能。

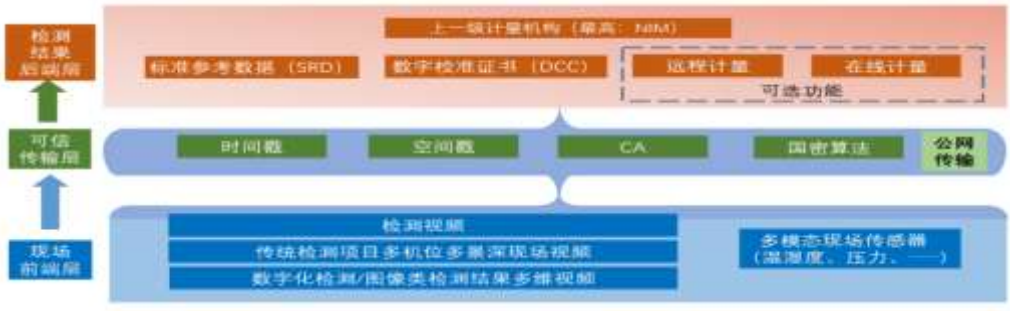

图 1 远程检测系统组成要素

5 可信视频的性能

可信视频是构成远程检测系统现场前端层的主要要素。为保证视频的可信传输，可信视频的格式应满足以下要求：

a）视频码流格式为H264、H265、SWAC的一种或几种；

b）视频必须在码流层进行可信编辑；

c）可信编辑的内容包括但不限于加密、解密、数字签名、数字验签、明文时间戳、暗文时间戳、明文空间戳、暗纹空间戳等。

6 远程检测系统的性能

6.1 真实性

主要指检测活动主体、客体的身份真实。真实性的具体指标主要包括以下要素：

a）主客体具备CA；

b）主客体的公钥、私钥齐备。

6.2 可信

主要指检测内容没有被篡改，必须是检测的所有要素都没有被篡改。可信的具体指标主要包括以下要素：

a）检测内容进行了加密；

b）加密的检测内容可以被解密；

c）检测内容进行了数字签名；

d）数字签名的检测内容可以被解签验签；

e）检测内容包含了明文时间戳、暗文时间戳的一种或两种；

f）检测内容包含了明文空间戳、暗纹空间戳的一种或两种。

6.3 完整性

主要指远程传输的所有检测内容没有丢包。可以采用奇偶校验位的方法进行验证。

7 性能测评要求

7.1 功能性测评

是对系统完成所声称的远程检测的功能进行测评。功能测评注重使用安全性，即该远程检测系统是否存在主客体在使用过程中发生破坏检测行为的可能性。使用安全性具体指标见附录A。

7.2 安全性测评

是对系统中是否存在的某种破坏系统安全能力的问题、错误进行测评。对系统进行安全分析,以发现系统安全缺陷或违反系统安全规范的动作。例如：

a）跨站脚本攻击

攻击者向 Web 页面里面插人恶意 HTML系统,当用户浏览该页面时,嵌入到 Web 里面的 HTML系统会被执行，从而达到攻击者的特殊目的。

b）SQL注入

将恶意 SQL 命令插入数据库请求参数,并提交给数据库执行的攻击行为。

c）异常

导致程序中断运行的一种指令流。注:如果不对异常进行正确的处理,则可能导致程序的中断执行。

7.3 防作弊测评

是对远程检测系统是否存在作弊系统、后门程序进行测评。

作弊系统也称为调试系统或后门，可以进入了一个计算机程序来改变运行时的行为或设定的程序。后门程序又称特洛伊木马，其用途在于潜伏在电脑中，从事搜集信息或便于黑客进入的动作。

7.4 计量特性测评

是对远程检测系统中的计量特性进行测评。

计量特性是关于远程检测系统中涉及到的量值和单位的特性，包括是否正确使用和表述，是否符合国际单位制，是否符合计量通用修约规则，是否具备本体化条件，量值计算位数是否一致，计算结果是否准确等。

8 性能测评方法

8.1 黑盒测试

采用人工审查的方式（见附录B），针对用户界面，基于程序的业务逻辑和系统功能点，进行逐个排查可能存在的安全功能缺陷、防作弊、二次开发和计量特性问题。黑盒测试有两种不同的方法，一种是定向功能分析法，该方法主要是根据程序的业务逻辑来测评的，根据系统的相关功能，大概推测可能存在哪些漏洞。常见功能漏洞：（包括但不限于）程序初始安装、站点信息泄漏、文件上传、管理、登陆认证、权限管理、数据库备份恢复、找回密码、验证码；另一种方法是功能点人工测评，这是对系统某个或某几个重要的功能点进行人工测评，发现功能点存在的安全问题。功能点人工测评需要收集系统的设计文档、系统开发说明书等技术资料，以便测评人员能够更好的了解系统业务功能。由于人工测评的工作量极大，所以需要分析并选择重要的功能点，有针对性的进行人工系统测评。

8.2 白盒测试

采用工具测评与人工测评（人工系统测评、人工审查工具测评结构、人工抽取系统检查）相结合，针对具体的系统进行检测。使用工具进行系统审，需要导入系统到测试环境，按预定系统策略进行测评，过滤系统测评结果，将漏洞进行分组（按漏洞类型，按功能模块）。使用人工测评时，一方面是基于工具测

试的结果，分析和撰写相应的测试系统，结合人工检查找出工具发现的漏洞中不确定和存在误报的漏洞，另一方面是确认问题系统覆盖的范围，深入分析发现的漏洞，并分析漏洞在当前系统中的严重等级（也可按不同的标准如：OWASP，PCI等）和系统引用的资源文件。同样地，白盒测试也有两种不同的方法，一种是整体系统测评，测评人员对被测评对象的所有系统进行整体测评，系统覆盖率为100%，这是一种最麻烦但也最全面的测评方法。整体系统测评属于白盒静态分析，仅能发现系统编写存在的安全漏洞，无法发现业务功能存在的缺陷；另一种是敏感函数参数回溯法（shell_ exec），即根据敏感函数逆向追踪参数传递的过程，大多数系统漏洞的产生是因为函数的使用不当导致，只要找到这样一些使用不当的函数，就可以快速挖掘对应的漏洞。

8.3 方法的选择

黑盒测试和白盒测试的不同点在于，黑盒测试是用户级测评，白盒测试是系统级测评，但其测评思路与方法具有相似性，在具体测评过程中，可以按照不同的问题选用更为合适的测评方法，对于较为复杂的系统，采用黑盒测试和白盒测试相结合，能够得到更为全面的测评结果。

测评时，根据测评对象和应用场景的不同，制定具体的测评方案，分配专业的系统测评人员和测评工具对系统进行测评，形成测评报告，并对测评出的问题与标准相关测评项逐一人工核对。测评实施过程中，可根据测评工作需要划分工作阶段。如按进度或里程碑划分;按周、月、季度划分;按功能模块实施单元划分;按人员分工交叉测评划分等。

对于使用外部开源系统较多的系统，在测评时可先检测开源系统的使用率，开源系统的安全缺陷可从已知漏洞角度检查。由于测评工具的局限性，不可避免存在误报和漏报。对于误报问题，应采用人工对比测评核查的方式开展。对干漏报问题应采用多个工具交叉测评的方式开展。人工测评是工具测评的必要补充，人工测评主要解决工具测评的误报和漏报问题。在人工测评实施中，可借助工具对系统模块、数据流、控制流等逻辑结构进行分析提取，并逐条比对分析。

9　性能测评流程

9.1 测评基本流程

测评过程包括四个阶段：测评准备、实施测评、人工测评、测评结果。 测评准备阶段，主要开展基本情况调研、签署保密协议、准备检查清单等工作；实施测评阶段，主要开展资料检查、信息收集、系统审查、结果分析等工作；人工测评阶段，主要开展计量特性测评、验证漏洞分析、定级风险等；测评结果阶段包括测评结果的总结、提交测评报告等工作，如有必要进行相关问题的澄清和相关资料说明，改进跟踪工作由系统开发团队进行，主要对测评出的问题进行修复。对于安全缺陷系统修改后，再次进行测评。测评流程（见图2）。本作业指导书提出系统测评的一般流程，在具体执行过程中根据对象或背景有所不同可适当增删。

收到系统文件及文档后，应首先进行病毒杀查，确定安全后再进行测量。

实施测评入场环节中，测评人员和项目成员（关键系统开发人员等）均应参与。测评人员介绍测评的主要目标、访谈对象和检查的资料等。项目人员介绍项目进展、项目关键成员、项目背景、实现功能以及项目的当前状态等。

信息收集环节主要通过访谈等方式获得系统以及相应需求分析文档、设计文档、测试文档等资料。通过文档资料了解系统的业务逻辑等信息。在了解系统基本信息的基础上，通过深人分析设计文档、访谈关键开发人员等方式，区分核心系统和一般性系统，其中核心系统一般为涉及核心业务功能和核心软件功能的系统，一般性系统为非核心业务功能和非核心软件功能的系统。

系统安全检测环节是根据制定的系统安全的检查项，采用工具测评、人工测评、人工结合工具测评方式检查是否存在安全缺陷，检测完成后进行安全性分析形成安全测评结果。

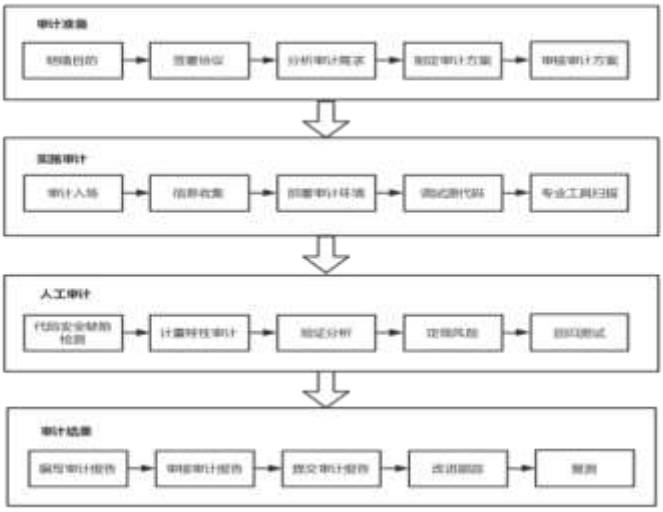

图 2　系统测评流程图

9.2 测评准备

测评准备包括以下几项。

明确测评目的：包括软件采购 / 外包测试、软件产品的认证测试、公司软件系统安全性自查等。

签署保密协议：为避免被测评单位的系统被测评方用于非系统测评用途，双方应签署系统测评保密协议，明确双方的权利和义务。

背景调研：了解系统的应用场景、目标客户、开发内容、开发者遵循的标准和流程等。

熟悉系统：通过阅读系统，了解程序系统结构、主要功能模块，以及采用的编程语言。

制定检查项目列表：通过明确测评目的、背景调研、熟悉系统等工作，形成系统安全测评要点，制定系统安全的检查列表。检查列表包括检查项和问题列表。

9.3 确定对象及范围

确定测评对象及其范围，测评对象是源系统（不限制语言）。

9.4 选取特性及指标

系统测评特性包括安全、防作弊、二次开发功能、计量特性，针对选取的测评特性要确定相应的测评指标。

9.5 确定具体指标及方法

系统测评在确定其对象后，应该根据测评特性的特点，确定其指标及测评方法，对于不同的测评特性一般是存在不同的指标的，以及需要不同的测评方法支持。考虑到测评内容的复杂性，方法采用工具测评和人工测评相结合，黑盒测试和白盒测试两种手段综合运用。

9.6 确定结果

测评完成后要对测评结果进行分析：

a）对测评目标与结果进行对比分析，确定是否得到正确的测评结果；

b）对测评的方案的有效性进行分析，确认是不是合适等。

测评实施完成后，组织召开评审会，将初始测评结果提供给被测评项目成员，并提供澄清误解机会，允许项目成员提供其他需要补充的信息。

评审会结束后，根据评审意见，调整测评结果，形成测评报告。测评报告包括测评的总体描述、测评结论等内容，并对可能产生的风险等级进行高、中、低分类描述。具体见附录C。

9.7 报告结果

系统测评报告是所有系统测评项目及其评测结果的合集。

在完整的系统测评报告中，应该包括但不限于以下内容：

a）测评依据的文件

b）测评范围

c）测评结论

附 录 A
（资料性）
功能性测评（使用安全性）参考指标

A.1 重定向

测评指标：不应开放不可信站点的URL重定向。

测评人员应检查系统是否存在URL重定向到不可信站点的情况，因重定向到不可信站点，可能会发生访问安全风险。

推荐用语：重定向错误、重定向异常。

推荐方式：人工测评（黑盒测试，用户级）

举例：验收管理点击订单详情，重定向异常。

A.2 身份鉴别过程中暴露多余信息

测评指标：应避免在处理身份鉴别的过程中暴露多余信息。

测评人员应检查账号在注册或认证过程中，是否存在暴露多余信息的情况。 攻击者可能会利用获取到的多余信息，进行认证暴力破解。

推荐用语：多余信息暴露。

推荐方式：人工测评（黑盒测试，用户级）

举例：网站在登录过程中显示密码的长度、强度等，易造成多余信息暴露。

A.3 配置信息

测评指标：宜对重要的配置信息进行安全保护。

测评人员宜检查系统中使用的重要配置信息是否进行了安全保护，因对重要配置信息保护不当，可能带来信息泄漏安全风险。

推荐用语：配置信息、信息泄露、敏感数据。

举例：某网站的连接信息，如地址、用户名和密码，没有对敏感数据加密保存造成信息泄露。

A.4 身份鉴别被绕过

测评指标：应避免身份鉴别被绕过。

测评人员应检查系统中身份鉴别机制是否存在被绕过的路径或通道，鉴别算法的关键步骤是否被省略或跳过。

推荐用语：身份鉴别机制、注入漏洞。

推荐方式：人工测评（黑盒测试，用户级）

举例：登录...系统，无需身份鉴别机制，存在注入漏洞，输入任何用户名与密码都可以进行入该系统。

A.5 身份鉴别尝试频率限制

测评指标：应对身份鉴别连续多次登录失败频率进行限制。

测评人员应检查系统中是否实现对身份鉴别多次登录失败的频率进行限制。 如结果为否定，则系统存在身份认证被暴力破解的安全风险。

推荐用语：频率限制。

推荐方式：人工测评（黑盒测试，用户级）

举例：登录...系统，多次显示登录失败仍可进行尝试登录操作，未设置针对登录失败次数的频率限制，存在安全风险。

A.6 口令安全

测评指标:应确保登录过程中口令不可明文显示，避免明文存储口令，避免明文传递口令。

测评人员应检查系统中是否实现在登录过程中口令是否明文显示，系统中是否存在明文存储口令的情况，系统中是否存在明文传递口令的情况。

推荐用语：是否存在...，应避免.....。

推荐方式：人工测评（黑盒测试，用户级）

举例：登录某网站时，用户登录过程中的口令以明文显示，存在安全风险。

A.7 权限管理

测评指标:应确保权限管理安全以及其他访问控制措施的安全。

测评人员应检查系统中的权限与访问控制功能相关部分,包括但不限于:是否缺失认证机制,如果结果为肯定,则提示存在安全风险；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；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；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。

推荐用语：权限...，存在不合理分配...,存在....。

推荐方式：人工测评（黑盒测试，用户级）

举例：某系统的角色管理模块中，申请人账户登录时可以停用“领导人”角色，权限管理功能存在隐患。

附 录 B

（规范性）
人工测评数据集

人工测评数据集见表 B.1。

表 B.1 人工测评数据集

分类	关键性	检验项	备注
命名			
	关键	命名规则是否和所采取规范保持一致？	组员变量，方法参数等需要使用首字母小写，其它单词首字母大写命名方法，严禁使用下划线(_)数字等方法命名不要出现局部变量，组员变量大写字母开头等问题
	通常	是否遵照了最小长度最多信息标准？	多种命名尽可能短，表意正确，除 2 替换‘to’4 替换‘for’外，不提议使用数字在命名中
	关键	has/can/is 前缀函数是否返回布尔型？	组员变量，方法参数，局部变量等为布尔型时假如出现 has/can/is 开头，将这些词去掉
	关键	类名是否存在重名问题？	自己实现类尽可能不要和它类重名，尽管不在同一个包下，尤其子类和父类重名情况
注释			
	关键	注释是否较清楚且必需？	方法 JAVADOC 注释中需要说明各参数、返回值及异常说明，参数说明需根据参数名称及用意对应标注
	关键	复杂分支步骤是否已被注释？	
	通常	距离较远是否已经被注释？	
	关键	函数是否已经有文档注释？（功效、输入、返回及其它可选）	文件，类（含接口，枚举等），组员变量，方法前需要有 JAVADOC 注释

表 B.1（续）

	通常	特殊使用方法是否被注释？	
申明空白缩进			
	通常	每行是否只申明了一个变量？（尤其是那些可能犯错类型）	
	关键	变量是否在定义同时初始化？	
	关键	类属性是否全部实施初始化？	
	通常	代码段落是否被以空行分隔？	
	通常	是否合理用空格使程序清楚？	基础代码格式中空格符不可缺乏，这些空格出现在?, :, +, -, *, /, =, ==, >, <, >=, <=, !=, 及多种括号周围
	提醒	代码行长度是否在要求之内？	每行不得超出 120 个字符
	关键	Controller，service,dao 中不要申明有状态变量	此变量不能被修改。假如要进行修改，必需经过锁进行控制
	通常	折行是否合适？	
	通常	集合是否被定义为泛型类型？	定义集合时，提议定义其泛型类型，降低类型转换和警告错误
语句功效分布规模			
	通常	包含复合语句{}是否成对出现并符合规范？	

表 B.1（续）

	关键	是否给单个循环、条件语句也加了{}？	if,else,else if,while,for,case 等代码块必需用{}包围
	通常	单个变量是否只做单个用途？	
	关键	单行是否只有单个功效？（不要使用；进行多行合并）	
	关键	单个函数是否实施了单个功效并和其命名相符？	
	通常	操作符＋＋和——操作符应用是否符合规范？	
规模			
	关键	单个函数不超出要求行数？	
	关键	缩进层数是否不超出要求？	
可靠性：总则变量语句			
	关键	是否已经消除了全部警告？	开发工具警告
	关键	常数变量是否申明为 final？	
	关键	对象使用前是否进行了检验？	
	关键	组员变量，局部变量是否在使用前被赋值？	对象初始化为 null 对象被调用前必需被重新赋值，假如赋值语句在 try 块中，调用操作必需在 try 块中
	通常	局部对象变量使用后是否被复位为 NULL？	尤其是数组，集合，Map

表 B.1（续）

	关键	对数组访问是否是安全？（正当 index 取值[0，MAX_SIZE-1]）。	
	关键	是否确定没有同名变量局部反复定义问题?	严禁局部变量名称和类或对象组员变量同名
	通常	程序中是否只用简单表示式?	
	关键	是否已经用（）使操作符优先级明确化?	
	关键	全部判定是否全部使用了（常量==变量或常量.equals(变量)）形式?	常量放在比较符前能够有效降低比较符写成赋值语句，降低空指针异常
	关键	是否每个 if-else if-else 语句全部有最终一个 else 以确保处理了全集?	
	关键	是否每个 switch-case 语句全部有最终一个 default 以确保处理了全集?	
	通常	for 循环是否全部用了包含下限不包含上限形式（k=0; k<MAX）	
	关键	XML 标识书写是否完整，字符串拼写是否正确?	
	关键	对于流操作代码异常捕捉是否有 finally 操作以关闭流对象?	关闭前需要判定流对象是否为空
	提醒	退出代码段时是否对临时对象做了释放处理?	
	关键	对浮点数值相等判定是否是合适?	严禁使用==直接判定浮点数值通用方法

表 B.1（续）

	关键	是否对象比较全部用 equals？	对象（包含包装类）比较必需使用 equals，而不是使用==或!=操作
	关键	使用 equals 进行比较时是否确保比较两个对象类型一致？	equals 方法比较对象在对象类型确定前提下，提议是同一类型，比如 Integer 和""使用 equals 是不提倡
	通常	操作 Map 或 Properties 结构对象，用于传值时是否将 Key 定义为常量？	Session，Request 等对象 set Attribute，Get Attribute 方法 key 提议使用常量，不得手工输入字符串
	关键	是否在类型转换前确保了类型兼容？	除非明确确保对象类型
	关键	包装类做简单预算前是否确保非空？提议全部使用包装类。	包装类进行操作前，提议进行非空（null != xx）判定预防发生空指针异常
	关键	对象属性在使用前是否确保被正确赋值？	只读属性（只提供 get 方法组员变量）除非特意返回固定值，不然必需提供 set 方法或在其它方法调用时将其赋值
	关键	方法调用前是否有非空判定？	对参数非空判定必需出现在方法调用之前，不然说明前面可能造成空指针或后者判定是没有必需，非空判定，默认由调用者提供
	关键	非线程安全对象是否被正确确保线程安全？	DateFormat 实例 format 方法调用不是线程安全，类似情况不适合使用 static 定义，提议使用 Thread Local 方法实现， 参看 Unified Code Generator
	通常	相同用意组员变量是否使用了相同命名？	不一样实体 Entity、VO、BO 之间表示同一含义组员变量，提议使用相同名称，尽可能不要出现，有地方用 username，有地方用 user Name 这么情况
可靠性：函数			

表 B.1（续）

	关键	入口对象是否全部被进行了判定不为空？	
	关键	入口数据正当范围是否全部被进行了判定？	
	关键	是否对有异常抛出方法全部实施了 try...catch 保护？	
	关键	是否函数全部分支全返回值？	
	关键	int 返回值是否合理？（负值为失败，非负值成功）	
	通常	对于反复进行了 int 返回值判定是否定义了函数来处理？	
	通常	关键代码是否做捕捉异常处理?	
	通常	字典表定义是否用枚举，或有一个统一定义？	
	关键	是否对方法返回值对象做 null 检验，该返回值定义时是否被初始化?	
	关键	是否对同时对象遍历访问做了代码同时?	
	关键	是否确定在对 Map 对象使用迭代遍历过程没做增减元素操作？	Map 遍历时实施增减元素操作将抛出 Concurrent Modification Exception，对集合对象遍历时提议全部不要进行增减元素操作
	关键	线程处理函数循环内部是否有异常捕捉处理，预防线程抛出异常而退出?	

表 B.1（续）

	关键	原子操作代码异常中止，使用相关外部变量是否恢复先前状态?	
	关键	函数对错误处理是合适？	
	关键	异常捕捉后是否进行了日志统计或异常继续抛出？	异常捕捉后假如无法处理需要继续抛出，假如能够处理，提议将异常日志进行统计
	关键	是否结构方法中不调用目前对象结构方法	严禁在结构方法中 new 一个目前对象
可维护性			
	关键	实现代码中是否消除了直接常数？（用于计数起点简单常数例外）	
	关键	是否消除了造成结构模糊连续赋值？（如 a=(b=d+c)）	
	关键	是否正确使用了日志统计?	
	通常	是否有冗余判定语句？（如：if(b)return true; else return false;）	“if (b) return true; else return false;”==》“return b;”;严禁使用类似“if/while(表示式 == true)或 if/while(表示式 == false)”判定
	关键	是否把方法中反复代码抽象成私有函数?	
代码警告			
	通常	是否清除了多出导入包或类?	
	关键	是否清除了只定义未使用局部变量？	严禁局部变量被定义或初始化而未被使用，这种情况需要删除该局部变量

表 B.1（续）

	通常	是否将魔鬼数字改常量使用？	不直接用除-2,-1,0,1,2,3,4,5,6,7,8,9,10外数字，除另外数字需要定义常量使用
	提醒	常量定义是否为 static final 格式？	public/protected//private staticfinal Type TYPE，static 和 final 次序要保持一致
	提醒	实现序列化对象是否定义了 serial Version UID?	提议实现 Serializable 类需要增加“private static final long serial Version UID=1L;”
可读			
	通常	是否用 if else 结构替换了三元运算符？	表示式复杂情况下不用 flag?exp1:exp2)语句，该语句需要修改为 if else 结构
	通常	代码注释率是否余 30%-60%间？	代码注释率应落在 30%-60%之间
性能			
	关键	日志统计 Log,Logger 对象是否定义为常量？	用于统计日志 Log,Logger 对象在类中定义必需是 static final，提议定义为 private，因为这类对象初始化较耗时不利系统运行
日志			
	关键	打印信息是否全用日志管理？	代码中提议不要使用 System.out.println 打印信息，只有在系统开启或系统立即退出时使用，其它部分全部用日志统计
圈复杂度			
	关键	单个类行数是否小于 500 行？	单个类提议行数小于 500 行最多不超出 1000 行
	关键	方法参数个数是否在 7 个以内？	方法参数个数提议小于 5 个，最多不超出 7 个

表 B.1（续）

	关键	单个方法函数是否小于 30 行？	单个方法提议函数小于 30 行，做多不超出 60 行
	关键	方法 try/for/while/switch/if 最深深度是否小于 5？	单方法中 try/for/while/switch/if 最深深度不许可大于 5
	关键	方法调用最深深度是否小于 15？	方法内部+内部调用累计深度不许可大于 15
SQL 空格			
	通常	连接符 or、in、and、和＝、<=、>= 等前后加上一个空格。	
	通常	逗号以后必需接一个空格。	
	通常	关键字、保留字和左括号之间必需有一个空格。	
SQL 注释			
	关键	对较为复杂 SQL 语句加上注释，说明算法、功效。注释风格：注释单独成行、放在语句前面。	
	关键	对关键计算应说明其功效。	SQL 中尽可能少包含业务逻辑
	通常	可采取单行/多行注释。（-- 或 /**/方法）。	
SQL 优化性能提议			
		1 书写 SQL 语句优化细则	

表 B.1（续）

	关键	1)尽可能避免相同语句因为书写格式不一样，而造成数次语法分析。	
	关键	2)多表连接时，使用表别名来引用列。	提议最多 5 个连接
	关键	3)不要在代码中用 SELECT *。	
	关键	4)where 条件中尽可能降低使用常量比较，改用参数变量。	
	关键	5)尽可能少用嵌套查询。请用 not exist 替换 not in 子句。	
	关键	6)用多表连接替换 EXISTS 子句。	
	关键	7)使用 UNION ALL 提升性能 。	
	关键	8)in、or 子句常会使用工作表，使索引失效；假如不产生大量反复值，能够考虑把子句拆开；拆开子句中应该包含索引。	
		2 排序注意事项	
	关键	1)大量排序操作影响系统性能，所以尽可能降低 order by 和 group by 排序操作。如必需使用排序操作，请遵照以下规则：	
	关键	a.排序尽可能建立在有索引列上。	
	关键	b.如结果集不需唯一，使用 union all 替换 union。	
		3 选择索引注意事项	

表 B.1（续）

	关键	1)对于复合索引，SQL 语句必需使用主索引列。	
	关键	2)索引中，尽可能避免用 NULL。	
	关键	3)对于索引比较，尽可能避免使用 NOT=（!=）。	
	关键	4)查询列和排序列和索引列次序保持一致。	
		4 其它经验性规则	
	关键	1)任何对列操作全部将造成表扫描，它包含数据库函数、计算表示式等等，查询时要尽可能将操作移至等号右边。	

附 录 C

（规范性）

系统测评结果的判定参照表

系统测评结果的判定见表C.1。

表C.1系统测评结果的判定参照

等级	等级划分依据	建议
高风险	可通过漏洞，获得管理员权限，完全控制机器和应用系统；利用该漏洞能够实现文件上传建立目录，从中获取大量的数据库表结构信息或利用该漏洞发起的攻击直接引起应用系统服务器故障、响应异常或资金损失。或者导致访问该网站的用户信息外泄的	代码开发者更改代码或提供说明
	具有明显的无法说明用途的代码出口	
	无法提供二次开发相关文件	
	严重影响计算结果的计量特性，例如导致计算结果错误，导致判断结论错误	
中风险	可通过漏洞，获取部分访问权限，造成内存泄露，利用该漏洞能够实现提取应用系统上相关信息，下载文件，但暂时无法实现文件上传或者建立目录。利用该漏洞攻击间接影响应用系统运行或引起客户敏感信息外泄	代码开发者更改代码或提供说明
	具有用途不清晰的代码出口	
	提供二次开发相关文件不全面	
	计量特性不规范，只影响计算结果的有效位数	
低风险	不安全的 Cookie 设置、无效的代码这些风险等级较低，不会对系统造成大的影响和危害的漏洞	不需要代码开发者做修改，必要时可提供说明
	具有冗余的代码出口	
	二次开发相关文件有表述模糊的部分	
	计量特性不规范，但不影响使用	