无标题-4

|  |  |  |  |
| --- | --- | --- | --- |
| 日志模块系统概要设计说明书 | | | |
| ■ 模板编号 | NSF-TR-RPM-103 | ■ 密级 | 公司机密 |
| ■ 模板版本 | V1.0 | ■模板发布日期 | 2013/10/31 |



|  |
| --- |
| © 2014 绿盟科技 |

|  |
| --- |
| ■ 版权声明 |
| 本文中出现的任何文字叙述、文档格式、插图、照片、方法、过程等内容，除另有特别注明，版权均属**绿盟科技**所有，受到有关产权及版权法保护。任何个人、机构未经**绿盟科技**的书面授权许可，不得以任何方式复制或引用本文的任何片断。 |

|  |  |  |  |
| --- | --- | --- | --- |
| ■ 版本变更记录 | | | |
| 时间 | 版本 | 说明 | 修改人 |
| 2014-5-19 | V0.1 | 创建 | 任鹏 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| ■ 适用性声明 |
| 产品整体，功能模块的概要设计 |

目 录

[日志模块系统概要设计说明书 1](#_Toc390155449)

[1 简介 5](#_Toc390155450)

[1.1 概述 5](#_Toc390155451)

[1.1.1 目的 5](#_Toc390155452)

[1.1.2 范围 5](#_Toc390155453)

[2 技术实现方案 5](#_Toc390155454)

[2.1 设计思路 5](#_Toc390155455)

[2.2 软件架构 6](#_Toc390155456)

[2.2.1 软件系统模块化视图（ Moudle View） 6](#_Toc390155457)

[3 概要设计 6](#_Toc390155458)

[3.1 Rest接口设计 6](#_Toc390155459)

[3.1.1 发送日志数据到sc 7](#_Toc390155460)

[1. 时序图 7](#_Toc390155468)

[2. 路径： 8](#_Toc390155469)

[3. 方法： 8](#_Toc390155470)

[4. 发送的数据格式： 8](#_Toc390155471)

[5. 返回的数据格式： 9](#_Toc390155472)

[3.1.2 从sc查询数据 10](#_Toc390155473)

[1. 时序图 10](#_Toc390155475)

[2. 路径： 10](#_Toc390155476)

[3. 方法： 10](#_Toc390155477)

[4. 数据格式： 10](#_Toc390155478)

[5. 返回的数据格式： 12](#_Toc390155479)

[3.2 Communication component接口设计 13](#_Toc390155480)

[3.2.1 功能说明 13](#_Toc390155481)

[3.2.2 事件处理流程 13](#_Toc390155482)

[1. 添加事件监听： 13](#_Toc390155483)

[2. 事件处理： 13](#_Toc390155484)

[3. 订阅推送 15](#_Toc390155485)

[3.2.3 订阅的匹配规则 16](#_Toc390155486)

[3.3 存储模块设计 17](#_Toc390155487)

[3.3.1 功能说明 17](#_Toc390155488)

[3.3.2 存储数据 18](#_Toc390155489)

[3.3.3 查询数据 19](#_Toc390155490)

[3.3.4 可测性/可维护性 20](#_Toc390155491)

[1. 保证功能的可测性 20](#_Toc390155492)

[*建议有可以覆盖所有功能或者部分功能的自动化测试工具* 20](#_Toc390155493)

[2. 保证模块的可调式性 20](#_Toc390155494)

[*建议能够方便的收集调试信息，模块出现问题时，方便根据自身的一部分信息来快速定位问题。提供*日志设计，包括审计，运行，调试日志等设计 20](#_Toc390155495)

[3.3.5 性能 20](#_Toc390155496)

[3.3.6 设计方案遗留的缺陷和限制 20](#_Toc390155497)

[3.4 数据库模块设计 21](#_Toc390155498)

[3.4.1 模块结构 21](#_Toc390155499)

[3.4.2 使用流程 21](#_Toc390155500)

[1. 如果要使用数据库存储数据，请首先实现接口IDBObject。主要是实现对表中字段的定义，完成对数据的序列化和反序列化过程。一个数据库表对应一个接口IDBObject的实现。 21](#_Toc390155501)

[2. 对于控制器的各个模块，如果需要数据库存储模块，请在获取模块依赖时，添加存储资源服务。详细如下： 22](#_Toc390155502)

[3. 获取存储模块服务： 22](#_Toc390155503)

[4. 初始化数据库表： 22](#_Toc390155504)

[5. 数据库表操作 22](#_Toc390155505)

[4 系统影响 23](#_Toc390155506)

[5 安全要素 23](#_Toc390155507)

[6 CBB通用部件 23](#_Toc390155508)

[7 技术风险和遗留问题 24](#_Toc390155509)

[8 工作任务包 和关联关系（可选） 24](#_Toc390155510)

[9 附件 24](#_Toc390155511)

# 简介

本概要设计文档描述SDN系统（日志管理模块）的概要设计。通过对其需求分析理解SDN系统（日志管理模块）的需求，并细化拆解核心功能描述其功能规格。并通过明确系统的用户可见的管理功能，说明系统的用户管理接口。经过需求分析定义后，提出技术实现方案，并对相关模块进行概要设计。通过系统影响章节描述本项目的设计对系统相关因素的影响。并描述系统的安全要素、通用部件、第三方软件等相关因素。

## 概述

### 目的

基于系统设计和项目需求文档，展开子系统的内部主要功能构成，模块划分，接口消息定义，以及重要的数据结构。同时依据功能构成的复杂度，关联性，以及已有软件的成熟度，对项目开发资源预估，工作计划进度制定提供科学判断基础。对于子系统的内部设计应该保持高内聚、低耦合，提升系统的软件扩展性，可维护性、安全性、软件、硬件的兼容性、模块之间的兼容性、以及性能和容量的扩展性、可测试性。

### 范围

SDN项目中日志模块模块和存储模块的设计都包含在此文档内，不包含其他模块的设计。

# 技术实现方案

*当有不同的替换方案是可以列举，不建议在概要设计文档评审阶段提供重大的可选替换方案。建议项目核心组工作对有重大影响的不同方案预研评估，早作抉择。*

*包含对功能性设计实现，以及系统设计中对系统影响要素的设计实现。*

*如果没有系统设计文档明确定义，需说明各子系统的实现语言。*

## 设计思路

 日志模块和外界的信息交互主要通过Rest接口和事件。Rest接口提供发送日志信息到日志模块和获取指定的日志信息内容。App和安全设备可以通过Rest接口发送日志信息。事件分为两种：普通事件和订阅。一般的情况，想获取日志信息的app或者其他模块可以通过订阅获取自己关心的内容。而其他模块如果想存储日志，则可以发送事件将日志信息附带在事件里面。

对于所有的日志信息都将存储到数据库中，对于某些设备的日志信息没有存储到本系统内的数据库的情况，请当产生日志的同时，给本模块发送一个消息或者post通知，告诉我有新的数据，然后日志模块到指定的数据库去读取数据。

## 软件架构

### 软件系统模块化视图（ Moudle View）

图2-1 日志模块内部结构图

# 概要设计

*High Level Design，这一节可以展开功能概述章节。* ***展开的粒度至少到相关子系统内部的相关重要功能模块，和关键的数据结构或状态机的定义。***

***以下子章节（包含功能构成图）是个示例可以作为参考，不同的产品/平台应该根据实际情况设定子章节及相关内容***

## Rest接口设计

*描述实现技术方案的细节实现，可以使用图片，流程图，图表等等。*

*尽量拆分到足够细，便于理解。可以扩张子章节 。不同的产品可以自行决定以下的子章节。*

*参考UML对设计进行描述：*

* 结构化描述（静态）
  + 类图对象图 描述类和他们的关系
  + 组件图 描述组件、接口和关系
  + Class responsibility collaborator cards (CRCs) 记录组件（类）的名字，职责和协作组件（类）的名字的头脑风暴工具
  + 部署图 节点和之间的关系
  + Entity-relationship diagrams (ERDs) 描述数据的概念模型
  + 结构图 用来描述程序的调用结构
* 行为描述（动态）
* 活动图Activity diagrams 描述活动间的控制流
* 协作图Collaboration diagrams 描述对象间交互，重点在对象及其间的关系，传递的消息
* 序列图Sequence diagrams 描述对象间的交互，重点在于时间顺序的传递的消息
* 数据流图Data flow diagrams (DFDs)描述进程间的数据流
* 流程图Flowcharts 描述控制流和执行的动作
* 状态图 描述状态机中状态间的控制流

### 发送日志数据到sc



#### 时序图

图3-1 安全设备或App发送日志到sc，sc推送日志消息时序图

#### 路径：

|  |  |  |
| --- | --- | --- |
| **上传的日志种类** | **路径** | **方法** |
| App日志 | /sc/logs/app/ | Post |
| 设备日志 | /sc/logs/dev/ | Post |

#### 发送的数据格式：

日志内容组成：Registerid、日志种类、日志类型、日志级别、时间、主机地址、对象类型、扩展字段。

1. Registerid：为日志上传者的注册id
2. 日志种类（category）说明见下表：

表3-1 日志类型说明

|  |  |
| --- | --- |
| **日志类型** | **说明** |
| AUDIT, | 审计日志 |
| INTERFACE\_MSG, | 接口消息日志 |
| SECURITY, | 安全日志 |
| SYS, | 系统日志 |
| RUN, | 运行日志 |
| SCAN, | 扫描日志 |
| ALERT, | 警告日志 |
| ATTACK, | 攻击日志 |
| Other | 其他日志 |

1. 日志类型（type）：主要是对日志种类的一个细分，比如说category为ATTACK, type可以为sql，ddos等
2. 日志级别（severity）：为数字，分别对应如下表

表3-2 日志安全级别说明

|  |  |
| --- | --- |
| **日志安全级别** | **说明** |
| 0 | Emergency: system is unusable |
| 1 | Alert: action must be taken immediately |
| 2 | Critical: critical conditions |
| 3 | Error: error conditions |
| 4 | Warning: warning conditions |
| 5 | Notice: normal but significant condition |
| 6 | Informational: informational messages |
| 7 | Debug: debug-level messages |

1. 时间：日志产生的时间。
2. 主机地址：发送者的ip地址。
3. objecttype：产生日志的对象类型。可以为：WAF、WVSS、NF等，或者xxxApp、FlowManager、PolicyManager、AppManager、LogManager、DeviceManager等
4. 扩展字段：消息实体。

所有的请求的消息格式为：

{

"head": {},

"data": {}

}

所有请求响应的消息格式为：

{

"opt\_status": code,

"head": {},

"data": {}

}

日志内容（detail）：日志消息举例。

|  |  |
| --- | --- |
| {  "head": null,  "data": {  "logs": [  {  "registerid": "140212121212",  "category": "ATTACK",  "type": "SQL",  "severity": 6,  "time": "2013-06-09 16:00:12",  "host": "10.65.110.80",  "objecttype": "WAF",  "hashid": "3289089082290",  "dst\_ip": "10.65.110.55",  "src\_ip": "10.65.222.11",  "content": "sql注入"  }  ]  }  } | {  "head": null,  "data": {  "logs": [  {  "registerid": "140212121212",  "category": "ATTACK",  "type": "SQL",  "severity": 0,  "time": "2013-06-09 16:00:12",  "host": "10.65.110.80",  "objecttype": "WAF",  "hashid": "3289089082290",  "dst\_ip": "10.65.110.55",  "src\_ip": "10.65.222.11",  "content": "sql注入"  },  {  "registerid": "140212121212",  "category": "ATTACK",  "type": "SQL",  "severity": 2,  "time": "2013-06-09 16:00:12",  "host": "10.65.110.83",  "hashid": "3289089082290",  "objecttype": "WAF",  "dst\_ip": "10.65.110.55",  "src\_ip": "10.65.222.11",  "content": "sql注入"  }  ]  }  } |
| **说明**：其中黑色字段为必填字段，蓝色字段为扩展字段，可以根据自己的业务选填。 | |

#### 返回的数据格式：

|  |  |
| --- | --- |
| 发送成功 | {  "opt\_status": 200,  "head": null,  "data": {  "result": "ok"  }  } |
| 发送失败 | {  "opt\_status": 404,  "head": null,  "data": {  " result ": "Error creating new report: Unexpected character "  }  } |

### 从sc查询数据



#### 时序图



图3-2 从sc查询日志数据时序图

#### 路径：

/sc/logs/query/

#### 方法：

post

#### 数据格式：

可选字段内容：日志种类、日志类型、日志级别、日志产生的时间、主机地址、对象类型、详细内容。如果不关心某个条件可以设为”\*”或者不填写。

1. 查询条件由字段名 + 比较内容组成。比较内容由比较操作符（可以缺省）和比较对象值组成。

比较操作符号如下所示

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EQ | LT | LE | GT | GE | AND | OR |
| 等于 | 小于 | 小于等于 | 大于 | 大于等于 | 与 | 或 |

注意：如果比较对象值为字符串，需要用单引号（’）包含起来，如果是数字则不需要。默认如果没有引号，则优先判断对象是否为数字，如果不是则确定为字符串或者其他类型。

例如：

"ATTACK", 或 "'ATTACK'" 表示为字符串

"severity": "GE 6 AND LE 10",

或 "severity":6

或"severity":"EQ 6"

或 "severity":"6"

都表示severity为数字类型。

"content": "EQ '78'", 或 "content":"'78'"

//表示匹配字段content值为字符串"78"。

1. 如果是详细内容里的查询条件，则需要查询者知道内容的具体存放格式，如果不确定字段内容，结果可能具有不确定性。
2. 一个字段内的属性比较操作如下表。多个规则之间禁止添加括号，例如：a AND b OR c
3. 字段之间属于与操作。

|  |
| --- |
| {  "head": null,  "data": {  "logs": {  **"category": "ATTACK OR SYS", //日志种类**  "type": "SQL OR DDOS", //日志类型  "severity": "GE 2 AND LT 5",  "time": "GT 2012-12-01 12:00:12 AND LT 2014-6-9 17:11:00", //时间范围  "host": "\*", //主机地址，如果为\*，可以不填写  "objecttype": "WAF OR FlowManager",  "dst\_ip": "10.10.1.1" //具体业务内容查询  }  }  } |
| 例如：  {  "head": null,  "data": {  "logs": {  "category": "ATTACK OR SYS",  "type": "SQL OR DDOS",  "severity": "GE 12",  "time": "GT 2012-12-01 12:00:12 AND LT 2014-6-9 17:11:00",  "objecttype": "WAF OR FlowManager"  }  }  } |

#### 返回的数据格式：

返回的数据字段包括：日志种类、日志类型、日志级别、日志产生的时间、主机地址、对象类型、详细内容

|  |  |
| --- | --- |
| 查询成功 | {  "opt\_status": 200,  "head": null,  "data": {  "logs": [  {  "category": "SYS",  "type": "SQL",  "severity": 12,  "time": "2013-06-10 04:00:12",  "host": "10.65.110.80",  "objecttype": "WAF",  "dst\_ip": "10.65.110.55",  "src\_ip": "10.65.222.11",  "content": "sql注入"    }  ]  }  或者  {  "opt\_status": 200,  "head": null,  "data": {  "logs": []  }  } |
| 查询失败 | {  "opt\_status": 404,  "head": null,  "data": {  "result": "Parse query json failuer."  }  } |

*由于暂时没有用到，暂不说明。*

## Communication component接口设计

### 功能说明

此模块主要负责接收事件和分发消息。接收外部的事件消息和订阅请求。App或者其他模块可以根据需要订阅自己感兴趣的部分，订阅条件满足后，LogModule就会将此模块的消息推送给订阅者。

### 事件处理流程

#### 添加事件监听：

监听普通的日志事件：RECEIVED\_LOG， 如果模块想要向日志模块发送一条日志信息，可以发送此事件。

监听订阅请求事件：ADD\_SUBSCRIPTION，如果模块需要订阅消息，可以发送订阅请求。

以上事件发送调用接口：void addEvent(Event e)

#### 事件处理：

1. 对于普通的event，重载IEventListener规定的事件通信接口：void processEvent(Event e)。在这里面对事件进行处理。此处主要处理接收到的日志存储消息。



图3-3 处理日志消息时序图

1. 对于订阅请求，重载IEventListener规定的事件通信接口：void processAddListenEventCondition(EventType type, EventSubscriptionInfo condition)。

为了防止模块重启后订阅者的请求信息丢失，对于每次接受的请求，序列化存储本地，当模块重启后，首先从本地加载上一次订阅者的信息，再监听新的请求，当有请求时更新列表和本地存储。



图3-4 订阅请求处理流程

#### 订阅推送

使用AppDataPusher模块，将信息推送给指定的url。



图 3-5 订阅推送处理流程

### 订阅的匹配规则

订阅的匹配条件包括：日志种类、日志类型、日志级别、主机地址、对象id、对象类型、详细内容字段等。各个条件具体格式见《3.1.1.4发送的数据格式》。

#### 增加订阅或者修改订阅

|  |  |
| --- | --- |
| 功能 | 新增或者删除 |
| Url地址 | /sc/apps/{appid}/subscription  其中appid为app注册id |
| 方法 | Post |
| 发送的数据格式 | {  "head": null,  "data ": {  "subscription": {  "name": "web\_attack\_subscribe\_1", //订阅规则的名称，可以用于修改该条规则或删除  "eventtype": "NEW\_REPORT\_ITEM", //订阅的时间类型  "callbackurl": "http://10.65.100.136:8888/sc/logs/test/", //推送的url地址  "filter": { //过滤规则  "category": "ATTACK OR ALERT",  "type": "SQL",  "severity": "GT 5",  "objecttype": "WAF OR NF"  }  }  }  } |
| 返回的数据格式 | {  "opt\_status": 200,  "head": null,  "data": {  "result": "ok"  }  } |

举例

|  |
| --- |
| {  "head": null,  "data": {  "subscription": {  "name": "web\_renpeng",  "eventtype": "NEW\_REPORT\_ITEM",  "callbackurl": "http://10.65.110.80:8888/sc/logs/test/",  "filter": {  "category": "ATTACK OR SECURITY",  "type": "WEBSEC OR SQL",  "severity": "GE 0",  "objecttype": "WAF OR NF"  }  }  }  }  {  "head": null,  "data": {  "subscription": {  "name": "web\_attack\_subscribe\_1",  "eventtype": "NEW\_REPORT\_ITEM",  "callbackurl": "http://10.65.100.136:8888/sc/logs/test/",  "filter": {  "category": "ATTACK OR ALERT",  "type": "SQL",  "severity": "GT 5",  "objecttype": "WAF OR NF"  }  }  }  } |

当有满足条件的数据时，app将受到如下格式的数据：

|  |
| --- |
| {  "opt\_status": 200,  "head": null,  "data": {  "**subscriptions**": [  {  "category": "ATTACK",  "type": "SQL",  "severity": 6,  "time": 1370764812000,  "host": "10.65.110.80",  "objecttype": "WAF",  "dst\_ip": "10.65.110.55",  "src\_ip": "10.65.222.11",  "content": "sql�e"  }  ],  "**subscribeId**": "web\_attack\_subscribe\_1"  }  } |

#### 删除订阅

|  |  |
| --- | --- |
| 功能 | 删除订阅 |
| Url地址 | /sc/apps/{appid}/subscription/{name}  其中appid为app注册id,name为订阅的名称。 |
| 方法 | Delete |
| 返回的数据格式 | **删除成功**  {  "opt\_status": 200,  "head": null,  "data": {  "result": "App[11403772450901][web\_renpeng] deleted succeed."  }  }  **删除失败**  {  "opt\_status": 404,  "head": null,  "data": {  "result": "The app id or subscriber name not found."  }  } |

日志模块根据以上六个选项，对日志进行匹配筛选，然后推送给订阅者。选项的内容可以为\*，表示匹配所有。每个选项的属性字段可以是一个数组。

## 存储模块设计

### 功能说明

为了应对日志数据种类多，格式无法统一的问题，采用Mongodb数据库，它对json具有良好的存储特性，无需关心其格式。

原始的日志json格式如下所示，除了detail项外其他的都是必须包含的字段。

|  |
| --- |
| {  “registerid”: “12378974645121”,  "category": "ATTACK",  //日志类型  "type": "sql",  //日志类型  "severity": 2,  //安全级别  "time": "2012-12-01 12:00:12",  //时间  "host": "10.65.110.80",  //主机ip  "hashid": 32890890890,  //安全设备id  "objecttype": "WAF",  //模块类型    //详细日志内容  *"dst\_ip": "10.65.110.55",*  *"src\_ip": "10.65.222.11",*  *"content": "sql注入"*  //以上日志字段内容，由安全设备自行任意填充，行为具有不确定性    } |

### 存储数据

为了适应日志格式的多样话，这里采用直接反序列化，即：将原始json字符串直接转化成DBObject，然后直接存储数据库，不做其他的格式化处理，从而兼容所有的日志格式。



图3-6 数据存储流程图

### 查询数据

由于本模块不关心日志的具体格式和内容，除了日志的必须字段一定存在，其他详细的日志内容中的某些字段，可能存在也可能不存在。使用者可以直接去查询，也可以在查询前可以先获取相关记录的所有字段名称，然后再作进一步操作。

查询的结果将以json的格式返回，与存入时的格式一样。



图3-7 数据查询流程

### 可测性/可维护性

#### 保证功能的可测性

#### *建议有可以覆盖所有功能或者部分功能的自动化测试工具*

#### 保证模块的可调式性

#### *建议能够方便的收集调试信息，模块出现问题时，方便根据自身的一部分信息来快速定位问题。提供*日志设计，包括审计，运行，调试日志等设计

### 性能

说明模块的性能对系统影响

模块本身的性能要求如：decoderl1\_4在系统中必须达到什么样的性能

模块的性能规定了之后，对后期系统整体的性能调优有一定的好处

### 设计方案遗留的缺陷和限制

设计方案实现需求过程中，如果存在一些缺陷和限制，记录在此。

## 数据库模块设计

### 模块结构

存储各类数据，模块的类图如下：

图3-8 数据存储模块类图

存储模块实现了ISecurityControllerService 和ISecurityControllerModule接口，同时扩展了自身的服务功能。具体的接口定义在IStorageSourceService。最终形成AbstractStorageSource接口。

对AbstractStorageSource的实现主要有3个，即StorageNotifyResource、MemoryStorageSource和MongoDBResultSet。它们实现对底层数据存储的具体操作，屏蔽差异化。这几个具体的实现，需要在启动模块时配置，指定具体哪一个服务被启动。

AbstractStorageSource接口还依赖于IDBObject接口，主要是实现对数据集的序列化和反序列化过程。当存储和查询获取数据的时候，需要对数据的具体格式进行解析，这时就要根据自己的业务需要，实现IDBObject接口，才能有效的将数据存储到数据模块中。

### 使用流程

#### 如果要使用数据库存储数据，请首先实现接口IDBObject。主要是实现对表中字段的定义，完成对数据的序列化和反序列化过程。一个数据库表对应一个接口IDBObject的实现。

public class CLASSNAME implements IDBObject{

Map<String,Object> getDBElements();

Object getFieldValueByKey(String key);

/\*\* This method must be implemented by the client of the storage API

\* to map the current row in the result set to a Java object.

\*

\* @param resultSet The result set obtained from a storage source query

\* @return The object created from the data in the result set

\*/

Object mapRow(IAbstractResultSet resultSet);

}

#### 对于控制器的各个模块，如果需要数据库存储模块，请在获取模块依赖时，添加存储资源服务。详细如下：

@Override

public Collection<Class<? extends ISecurityControllerService>> getModuleDependencies() {

Collection<Class<? extends ISecurityControllerService>> l = new ArrayList<Class<? extends ISecurityControllerService>>();

l.add(IStorageSourceService.class);

l.add(IRestApiService.class);

}

#### 获取存储模块服务：

this.storageSource = context.getServiceImpl(IStorageSourceService.class, this);

#### 初始化数据库表：

创建数据库表，设置主键等。

例如：

this.storageSource.createTable(tableName, tableColumns);

this.storageSource.setTablePrimaryKeyName(tableName, this.primaryKeyName);

#### 数据库表操作

所有的具体数据库存储驱动都要实现接口AbstractStorageSource，目前主要的驱动有：JDBCSqlStorageSource（关系型数据库）和MongoDBStorageSource（菲关系型数据库）**，**其中的主要接口如下：

|  |  |
| --- | --- |
| 函数名称 | 功能 |
| createTable(String tableName, Map<String, String> indexedColumns) | 创建表 |
| public void setTablePrimaryKeyName(String tableName, String primaryKeyName) | 设置主键 |
| public Set<String> getAllTableNames() | 获取所有的表 |
| public QueryClause createQuery(String tableName, String[] columnNames,  List<QueryClauseItem> clauseItems, RowOrdering ordering) | 创建查询条件 |
| public QueryClause createQuery(String tableName, String[] columnNames,  String condition, RowOrdering ordering) | 创建查询条件 |
| public int updateOrInsertEntity(String tableName, Object entityKey, IDBObject entity) | 更新或插入 |
| public int updateOrInsertEntity(String tableName, IDBObject entity) | 更新或插入 |
| public int deleteEntity(String tableName, Object entityKey) | 删除 |
| public int deleteEntities(String tableName, QueryClause query) | 删除 |
| public IAbstractResultSet executeQuery(QueryClause query) | 根据查询条件执行查询 |
| public IAbstractResultSet executeQuery(String tableName, String[] columnNames, String condition, RowOrdering order) | 根据查询条件执行查询 |
| public List<Object> executeQuery(QueryClause query, IDBObject rowMapper) | 根据查询条件执行查询 |
| public List<Object> executeQuery(String tableName, String[] columnNames, String condition, RowOrdering order, IDBObject rowMapper) | 根据查询条件执行查询 |
| public Object getEntity(String tableName, Object entityKey, IDBObject rowMapper) | 根据给定的id获取一行记录 |

# 系统影响

*是否存在知识产权纠纷*

# 安全要素

*所有的自身安全性问题*

# CBB通用部件

*概要设计列出系统中使用到的通用部件以及可以输出的通用部件*

# 技术风险和遗留问题

*描述概要设计方案的技术风险，以及遗留问题*

# 工作任务包 和关联关系（可选）

*定义实现设计所需要的开发工作任务包(Work package)，以及开发任务包的关联关系(Anatomy)。任务包可以是基于子系统，或多个子系统。*

*如果删除此章节，相关内容请使用其他内容进行体现。*专利

*描述系统中可申请专利的算法、设计等，待系统实现后，申请相应的专利。*

| **编号** | **专利名称** | **专利内容** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# 附件

*列举所有的，提到的，可能的附件，例如特定的消息定义等等*

## WAF日志说明：

|  |  |  |  |
| --- | --- | --- | --- |
| **category** | **告警事件** | **type** | **页面展示** |
| SECURITY | HTTP协议校验 | 1 | HTTP协议违背 |
|  | Web服务器漏洞 | 2 | WEB服务器漏洞攻击 |
|  | Web插件漏洞 | 3 | WEB构件漏洞攻击 |
|  | 爬虫防护 | 4 | 爬虫事件 |
|  | 扫描防护 | 5 | 恶意扫描 |
|  | 跨站请求伪造 | 6 | 跨站请求伪造 |
|  | 文件非法上传 | 7 | 文件非法上传 |
|  | 跨站脚本防护 | 8 | 跨站攻击 |
|  | SQL注入防护 | 9 | SQL注入攻击 |
|  | LDAP注入防护 | 10 | LDAP注入攻击 |
|  | SSI指令防护 | 11 | SSI指令攻击 |
|  | XPath注入防护 | 12 | XPATH注入攻击 |
|  | 命令行注入防护 | 13 | 命令注入攻击 |
|  | 路径穿越防护 | 14 | 路径穿越攻击 |
|  | 远程文件包含防护 | 15 | 远程文件包含 |
|  | 目录遍历防护 | 16 | 目录索引信息泄露 |
|  | 服务器信息泄露 | 17 | 服务器信息泄露 |
|  | 内容过滤 | 18 | 页面内容非法 |
|  | 非法下载限制 | 19 | 非法下载 |
|  | 页面篡改 | 20 |  |
|  | 自定义 | 21 | 自定义攻击 |
|  | ARP防护日志 | 22 |  |
|  | SYN\_FLOOD攻击 | 23 |  |
|  | ACK\_FLOOD攻击 | 24 |  |
|  | HTTP\_FLOOD攻击 | 25 |  |
|  | IP访问控制事件 | 26 | IP访问控制事件 |
|  | HTTP访问控制 | 27 | HTTP访问控制事件 |
|  | WEB访问日志 | 28 |  |
|  | 智能补丁 | 29 | 智能补丁 |
|  | 防盗链 | 30 | 资源盗链 |
|  | Cookie安全 | 31 | COOKIE篡改 |
|  | 白名单 | 32 | 违背白名单 |
|  | 例外策略 | 33 | 例外控制事件 |
|  | 联动事件 | 34 |  |
|  | WebShell防护 | 35 | Webshell页面访问 |
|  | 敏感信息泄露 | 36 | 敏感信息泄露 |
|  | 站点访问量统计 | 37 | 站点访问量统计 |
|  | 数据安全传输 | 38 | 不安全的数据传输 |