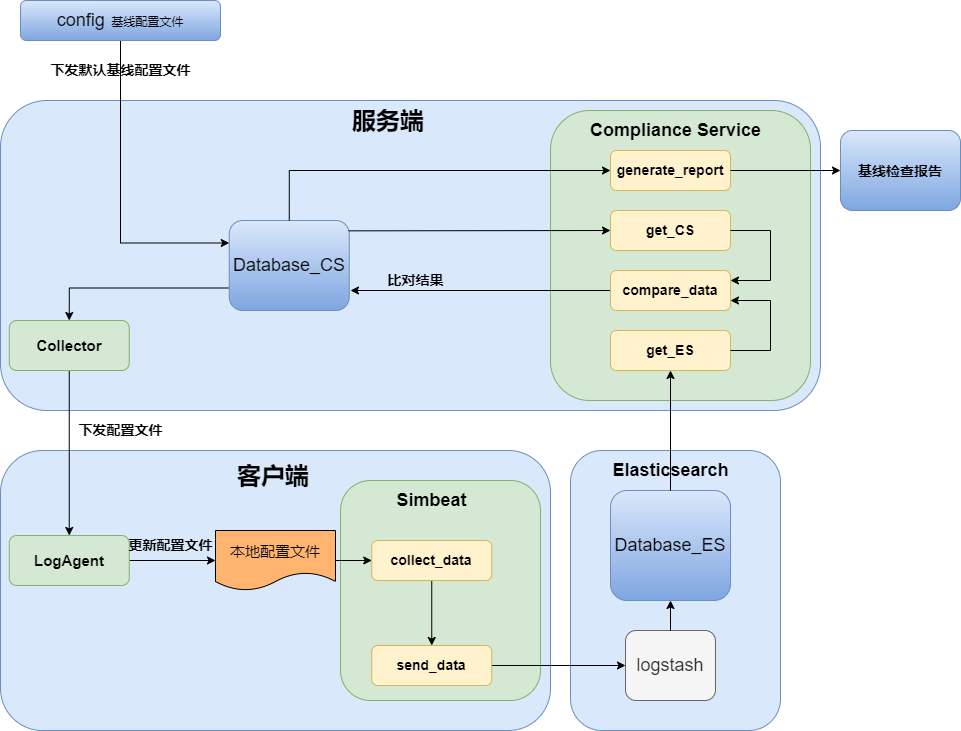
## 1.系统整体架构



系统总体可以分为两大模块，分别是读取系统信息的simbeat模块以及处理系统信息的Compliance Service模块。

**Simbeat模块主要包含的功能如下**：

1）根据本地的配置文件表定期查询机器的配置信息；

2）将查询到的数据经过logstash标记说明名字并过滤信息，然后保存到Elasticsearch中。

其中，LogAgent模块接收来自Collecttor的配置文件，并将其与存储并更新本地目录中的配置文件。

**Compliance Service模块主要包含功能如下**：

1）基线对比模块定期从Database\_ES和Database\_CS分别读取数据，通过RuleID逐条比对数据的预期值和查询值，将结果存入数据库中；

2）完成基线对比操作后读取数据库，根据比对结果生成基线检查报告存储至指定目录。

其中，配置文件直接下发存储到数据库中，然后Collector模块从数据库中读取特定的字段，组成简略版的配置文件，并将其同步到LogAgent中。

## 2.模块/功能设计

### 2.1 Simbeat模块

考虑到Simbeat需要运行在Windows10和Debian10两个操作系统中，两个操作系统的特性决定了信息采集的方式有较大的区别，因此本系统针对两个操作系统分别设计Simbeat模块。

#### 2.1.1 Debian10系统版本的Simbeat

##### 2.1.1.1 模块运行流程

Debian10系统中，通过LogAgent同步数据信息，Simbeat到指定目录中读取配置信息，然后根据配置信息，然后根据配置信息，循环读取每一个检查项对应的命令行，查询本机配置，并将命令行的结果上传到ES中即可，其具体流程如图2-1所示。

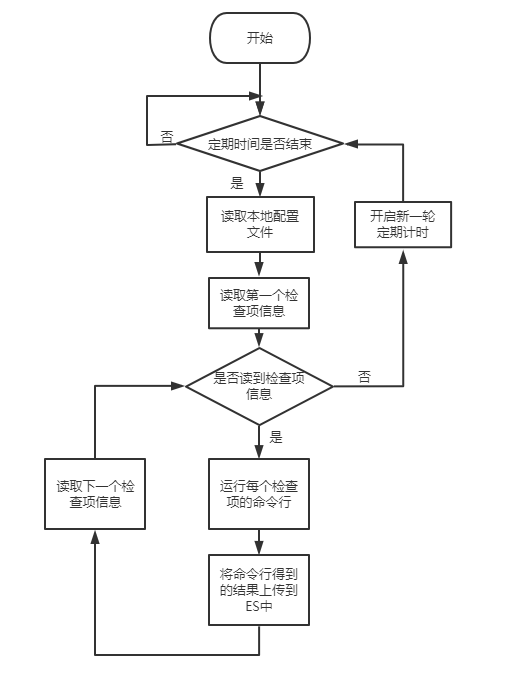


图2-1 信息采集上传流程

##### 2.1.1.2 数据接口设计

Debian10本地配置文件格式如下图2-2所示：



图2-2 Debain10本地配置文件格式

由图可知，本地文件中首先包含两个字段，其中ID表示该本地文件使用的是哪个版本的配置配置文件，而check是检查项需要的信息，其格式为以ruleID作为分隔体，每一个ruleID表示已一个检查项，里面包含内容，分别是该检查项的ID以及该检查项的检查参数，Debain10中由于每个检查项是通过执行命令行来获取系统参数结果，所以参数里面为每个检查项的命令行内容，只需要执行该命令该命令行即可获取结果，若一个检查项里面有多个内容，则只需通过逗号隔开即可，运行时会依次运行每个命令行并保存运行后的结果。

若需要更新和扩展系统的检查项，则只需要在数据库中更新配置文件，然后通过Collector发送扩展更新后的配置文件，然后LogAgent接收后更新本地的配置文件即可。

#### 2.1.2 Windows10系统版本的Simbeat

##### 2.1.1.1 模块运行流程

Windows10系统中，通过LogAgent同步数据信息,Simbeat只需到指定目录读取配置信息，然后根据配置信息读取本机配置并将检查信息上传到ES中即可，该部分可以分为三个部分，具体流程如下：

**（1）组策略存入数据库**

组策略的检查通过读取注册表的方式完成，由于其注册表路径包含随机数，所以单独拉取出来放入数据库中进行检索，其在数据库中的处理如下图所示。具体如图2-3所示。

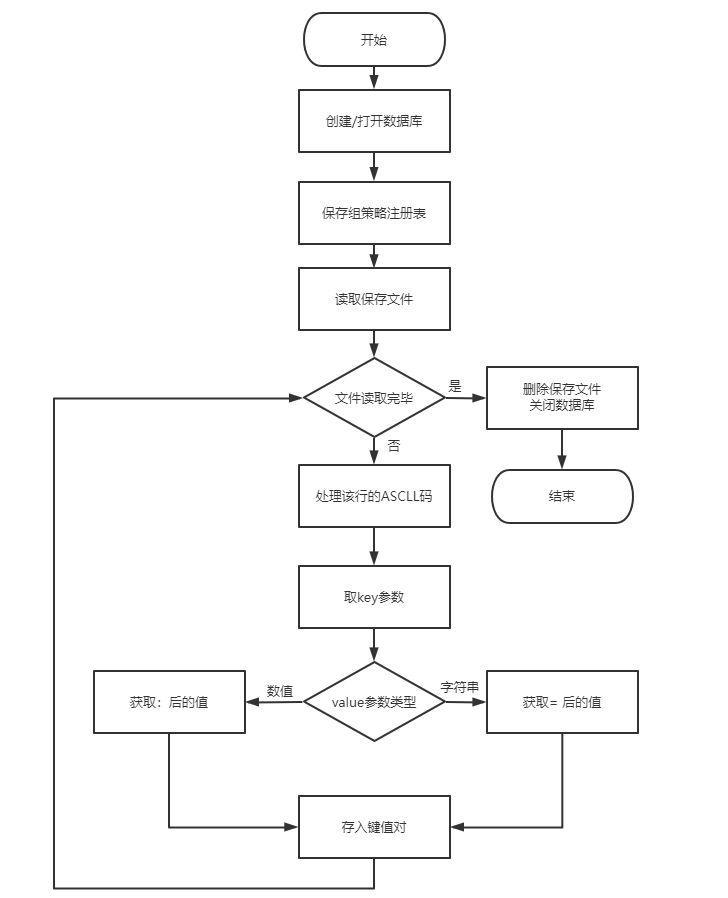


图2-3 组策略存储流程

**（2）服务存入数据库**

服务的检查通过 cmd 命令查看已开启的服务的方式完成，将已开启的服务名称作为 key 存入数据库中，其 value 均为 1，其在数据库中的处理如下图2-4所示。

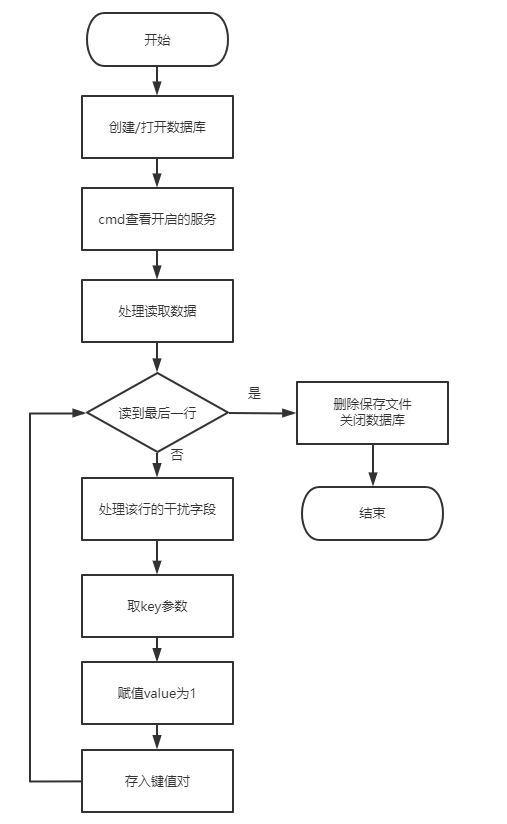


图2-4 服务存储流程

**（3）处理检查项并上传结果**

首先完成（1）和（2）中的存入数据库操作，再读取本地配置文件和基线配置文件，循环处理基线配置文件的每一个检查项，用 ruleID 匹配本地配置文件的检查参数作为检索数据库的参数，从而获得检查项结果，最后统一发送所有的检查结果。具体如下图2-5所示。

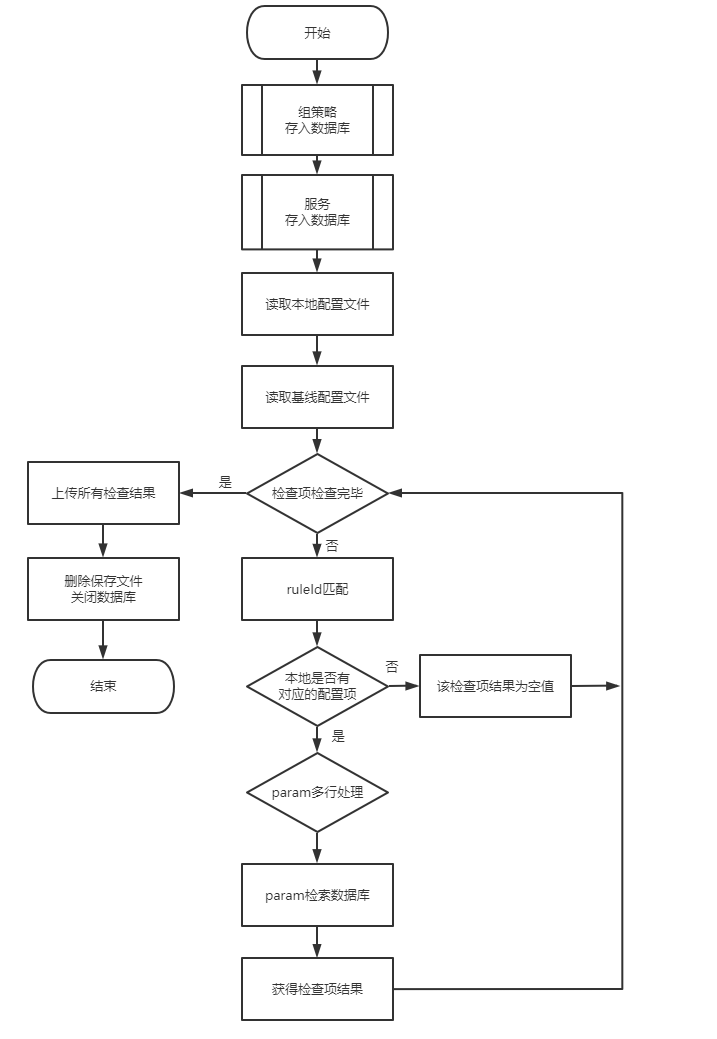


图2-5 信息采集上传流程

##### 2.1.1.2 数据接口设计

**（1）本地配置文件**

Windows10本地配置文件格式如下图2-6所示：



图2-6 Windows10本地配置文件格式

由图可知，本地文件中首先包含两个字段，其中ID表示该本地文件使用的是哪个版本的配置配置文件，而check是检查项需要的信息，其格式为以ruleID作为分隔体，每一个ruleID表示已一个检查项，里面包含内容，分别是该检查项的ID以及该检查项的检查参数。

Windows10中由于查询服务和组策略的方式均为在数据库中查找，所以参数的格式是固定的：

1. 服务的参数时使用GB18030 字符集的服务名称；
2. 组策略的参数为注册表中uuid后的路径+注册表键名，两者用冒号隔开。

若一个检查项里面有多个内容，则只需通过逗号隔开即可，运行时会依次运行每个命令行并保存运行后的结果。

若需要更新和扩展系统的检查项，则只需要通过compliance service发送扩展后的配置文件，然后beat接收后更新本地的配置文件即可。

**（2）本地数据库**

选择非关系数据库badger，其优势在于：

1. 以键值对的形式实现检查项的存储和搜索
2. 无需本地安装软件
3. 纯go语言实现，无需开启cgo模块

数据的存储方式为键值对，key是配置文件的param，value是相应检查点的值，也是最终需要上传的目标。

#### 2.1.3 接口设计

beat端存在两个配置文件，分别是上述包含查询信息的本地配置文件以及beat自带的配置文件，其中本地配置文件的格式如上图2-6,2-2所示，beat自带的配置文件的格式如下图2-7，2-8所示。由图可知，beat自带的配置文件主要配置一些beat自带的信息，本程序主要设置了上述数据配置文件的路径信息，每次查询的间隔时间，以及beat传送的elk的地址

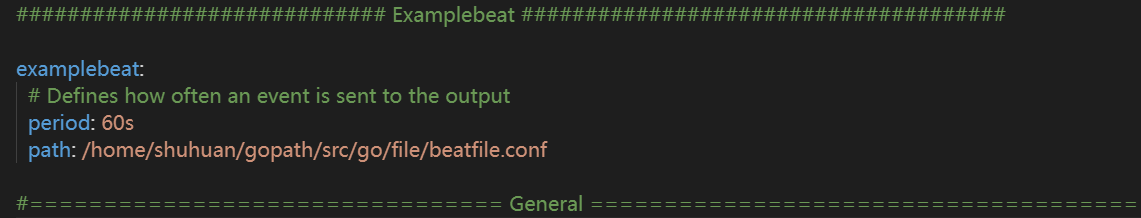


图2-7 配置文件图

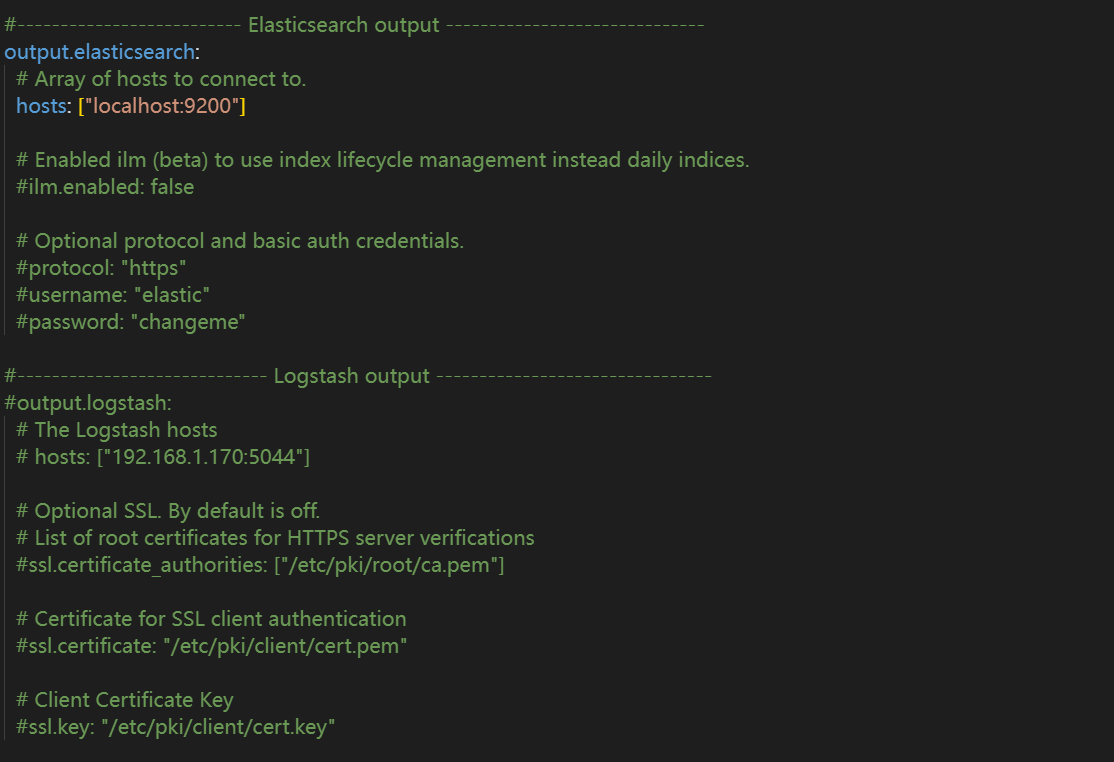


图2-8 配置文件图

### 2.2 Compliance Service

#### 2.2.1 基线配置文件采集/下发模块

##### 2.2.1.1 模块运行流程

配置文件直接下发存储至数据库中，其数据库的格式如表3-3所示，表中个字段的详细内容如下2.2.1.2所示，然后Collector模块可以根据数据库中的内容，抽取出其中一部分字段组成如上图2-2,2-6所示的简略版本，并将其同步到对应主机的LogAgent中。

##### 2.2.1.2 数据接口设计

Compliance Service中存储的基线配置文件，是整个系统的基线配置文件，所以其包含了Windows10和Debain10所需要的配置文件的集合，如下图2-7所示，由图可知，配置文件首先包含“Debain10”和“Windows10”两个分区，用于包含两个系统所需的配置文件，其中包含一个“checkpoint”的内容，其中存储每个检查项的配置内容，每个检查项以“ruleId”名作为区分，其中包含有该检查项的“ruleId”，该检查项的预估值“data”，该检查项结果的类型标记“type”值，若是常规直接比对的话，其值为“0”，若需要进行文本处理得到结果的话，其值为“1”，最后是“param”参数值，Debain10中其值为系统命令行运行是需要的命令行语句，Windows10中其值为Simbeat进行查询所需的path。

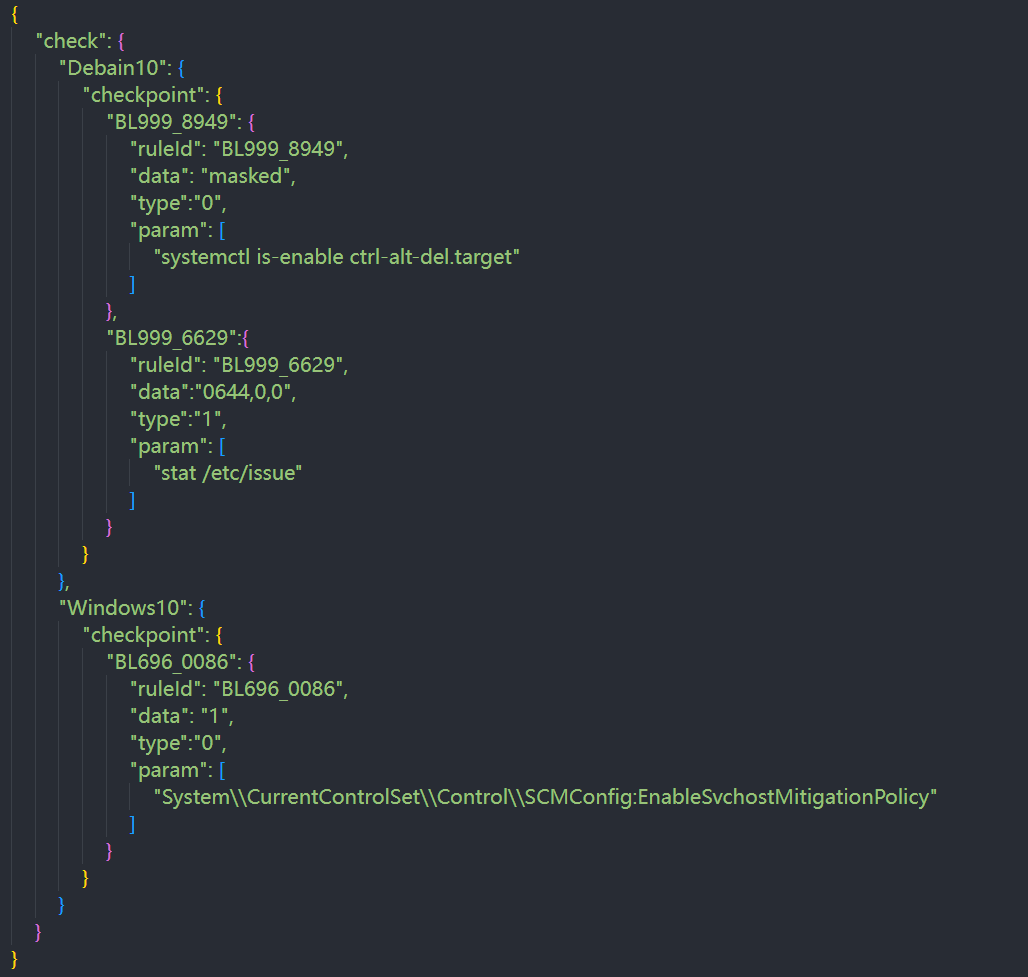


图2-7 配置文件格式

#### 2.2.2 定期基线比对模块

##### 2.2.2.1 模块运行流程

分别将存储在CS数据库中的配置文件和ES数据库中的结果读取出来，然后通过ruleId逐条比对CS数据库中预期值和ES数据库中的结果值，将结果存入检查清单，并根据比对的collectTime和hostip存入对应的统计结果，完成基线对比操作后根据统计结果，生成基线检查报告并存入数据库中。其具体的流程如图2-8所示。

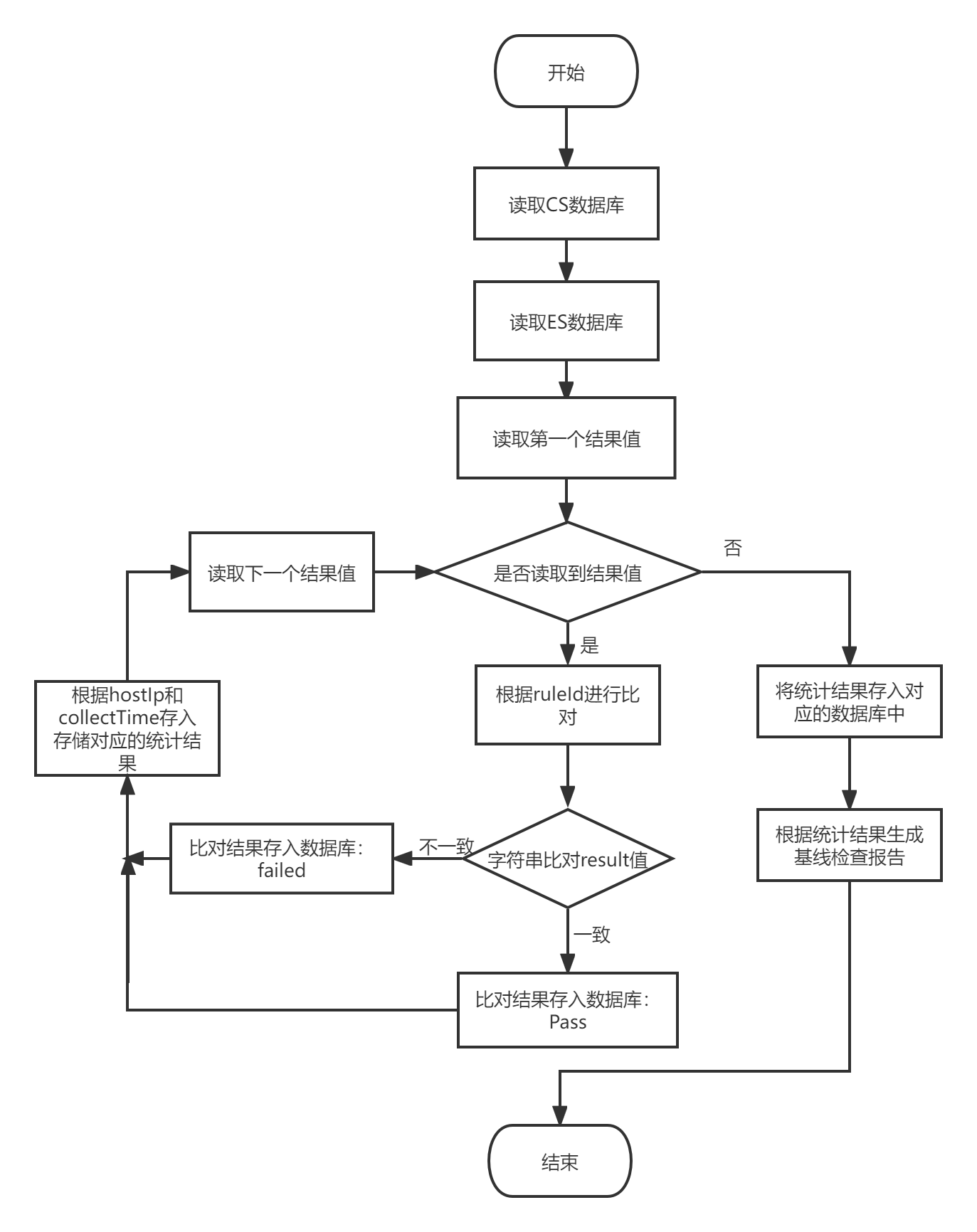


图2-8 比对模块流程图

##### 2.2.2.2 数据接口设计

上述流程中，根据最后的检查清单，会生成对应的基线检查报告，其格式如图2-9所示，由图可知，报告为一个json格式文件，其中包含四项，分别是表示检查机器系统的“OS”项，报告生成时间“time”项，标记比对主机的“hostIP”项，以及报告结果的“result”项，而“result”项里面包含有两个项，分别是比对正确的“true”项和比对失败的“false”项，里面存储本次别对的结果值，即预期值和实际值。



图2-9 基线检查报告格式

#### 2.2.3 报告生成模块

##### 2.2.3.1 实时性报告

**（1）全机读取**

该部分为生成所有机器的最新一次检测结果的报告数据，具体的实现流程是，读取数据库中的统计结果表，首先读出表中的所有主机ip，然后根据主机IP，依次读取数据库，取出每个ip中最新一次的比对统计结果，最后将该ip的统计结果数据整合成list形成结果报告。具体流程如图2-10所示。

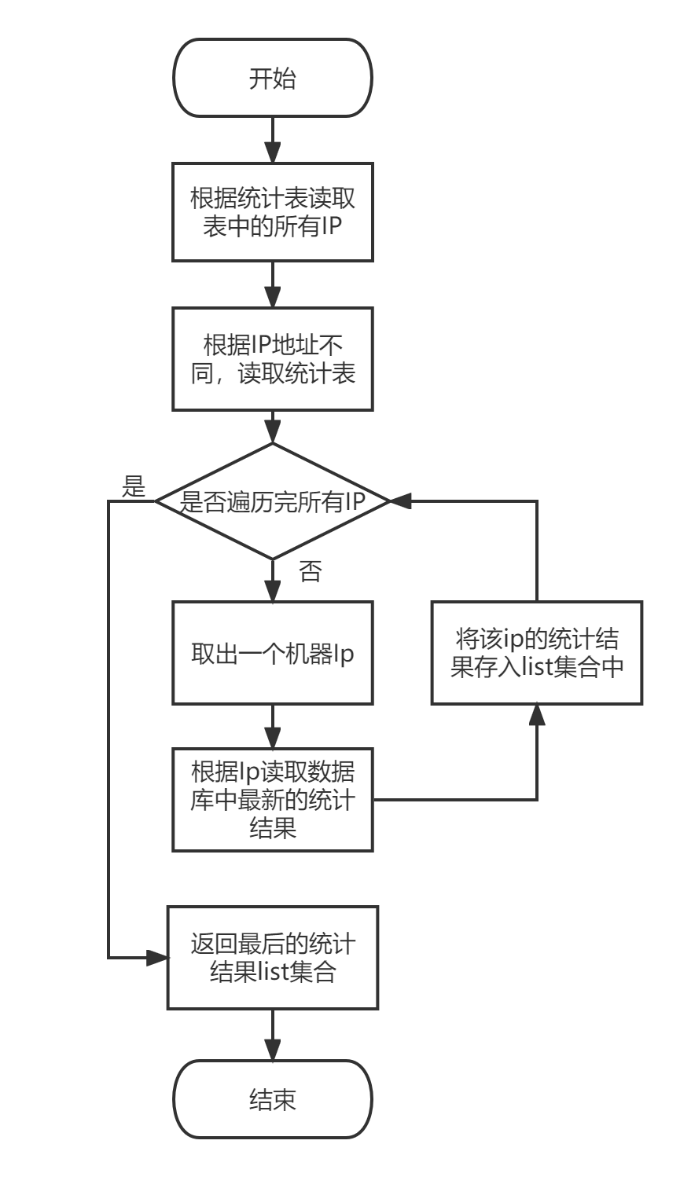


图2-10 全机读取实时性报告生成流程

上述全机读取流程中的输入输出格式如下表2-1所示：

表2-1 全机读取实时报告参数表

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 字段 | | 备注 |
| 输入 | 无 | | 无 |
| 输出 | List*<*hostResult> HostList | | 检查机器列表 |
| hostResult | checkTime | 最新检查时间 |
| hostIp | 检查的机器ip |
| failedList | 失败的检查项列表（存放ruleID） |
| passList | 成功的检查项列表（存放ruleID） |

**（2）IP读取**

该部分为根据IP生成部分机器的最新一次比对数据的结果报告，具体流程为通过输入IP地址，这个ip地址可以是单个ip，也可以是不连续的ip地址序列，同时还可以是ip地址段，然后根据读入的ip地址，对其进行解析，转换成对应的ip地址序列，然后循环遍历每个ip，根据ip读取数据库中相应的最新的统计结果数据，最后将这些ip的统计结果数据整合成list，形成最后的结果报告，具体流程如下图2-11所示。

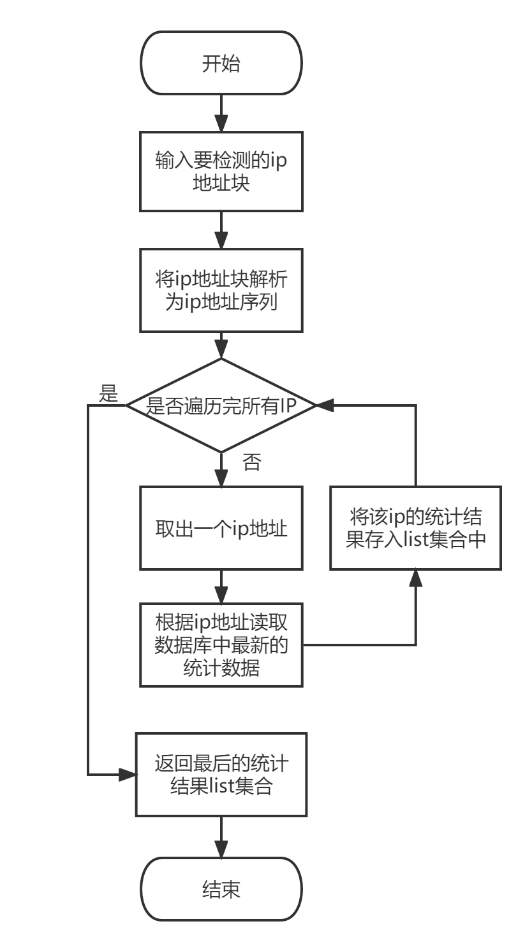


图2-11 ip读取实时性报告生成流程

上述根据ip读取流程中的输入输出参数如下表2-2所示。

表2-2 ip读取实时报告参数表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 字段 | | | 备注 |
| 输入 | hostIp | | 单ip | 单个ip地址 |
| 多ip | 多个ip地址序列 |
| ip地址块 | 如192.168.132.0/24 |
| 输出 | List*<*hostResult>HostList | | | 检查机器列表 |
| hostResult | checkTime | | 最新检查时间 |
| hostIp | | 检查的机器ip |
| failedList | | 失败的检查项列表（存放ruleID） |
| passList | | 成功的检查项列表（存放ruleID） |

##### 2.2.3.2 周期性报告

**（1）全机读取**

该部分用于生成一个周期时间段内，对于所有机器的长时间监测中，机器监测趋势的统计，具体的流程为，首先输入比对周期的时间范围，如开始时间和结束时间，以及比对报告的数据频度，如，以小时为单位，以天数为单位，以月份为单位等。然后读取统计数据库中的ip数据，然后根据数据的周期范围和数据频度，以数据范围边距，以数据频度为跨度，每次取数据频度中，每个ip的最新的一次统计结果数据，最后根据每次频度的即时结果存入最后的周期结果list中。具体实现流程如图2-12所示。

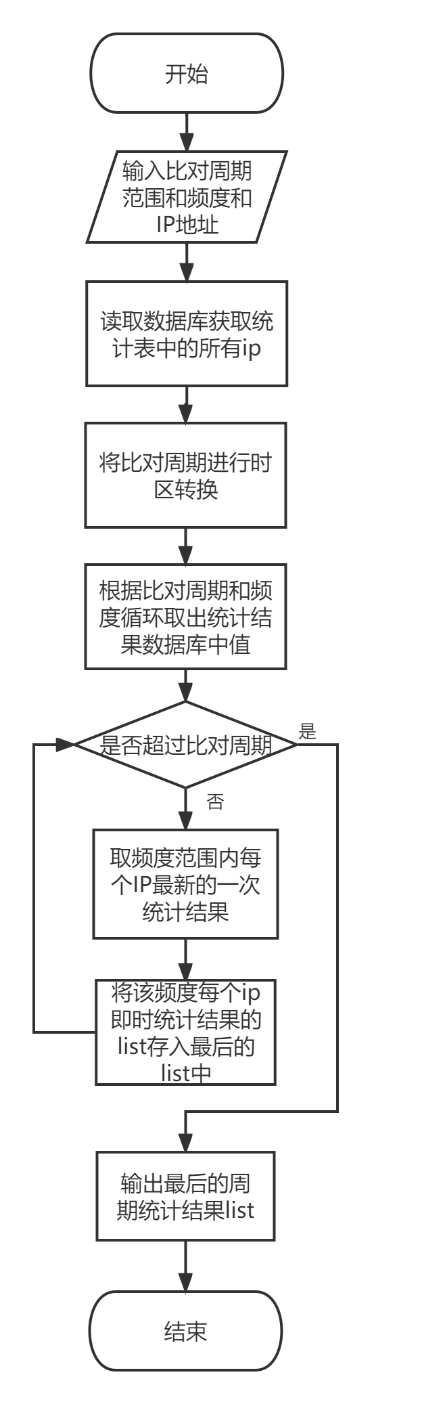


图2-12 全机读取周期性报告生成流程

上述全机读取的流程中，输入输出的参数格式如下表2-3所示。

表2-3 全机读取周期性报告参数列表

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 字段 | | 备注 |
| 输入 | startTime | | 周期开始时间，如  2022-07-14 00:00:00 |
| endTime | | 周期结束时间，如  2022-07-22 00:00:00 |
| utc | | 标记时区，如+8 |
| timeFrequence | | 时间频度，如year month day hour 等 |
| multiple | | 倍数，如1表示一个频度，如一天、一小时 |
| 输出 | ResultList | | 每个周期内最新检查时间的即时报告信息，具体格式如下 |
| List*<*hostResult*>* HostList | | 检查机器列表 |
| hostResult | checkTime | 最新检查时间 |
| hostIp | 检查的机器ip |
| failedList | 失败的检查项列表（存放ruleID） |
| passList | 成功的检查项列表（存放ruleID） |

**（2）ip读取**

该部分为对部分机器，在长时间的检测比对中，生成对应的周期报告，具体流程为可以通过输入IP地址，这个ip地址可以是单个ip，也可以是不连续的ip地址序列，同时还可以是ip地址段，然后根据读入的ip地址，对其进行解析并转换成对应的ip地址序列，然后根据比对周期和频度，在每个区间段内，对每个ip地址，读取其在该时间段内最新的统计数据，最后根据每次频度的即时结果存入最后的周期结果list中，具体如图2-13所示。

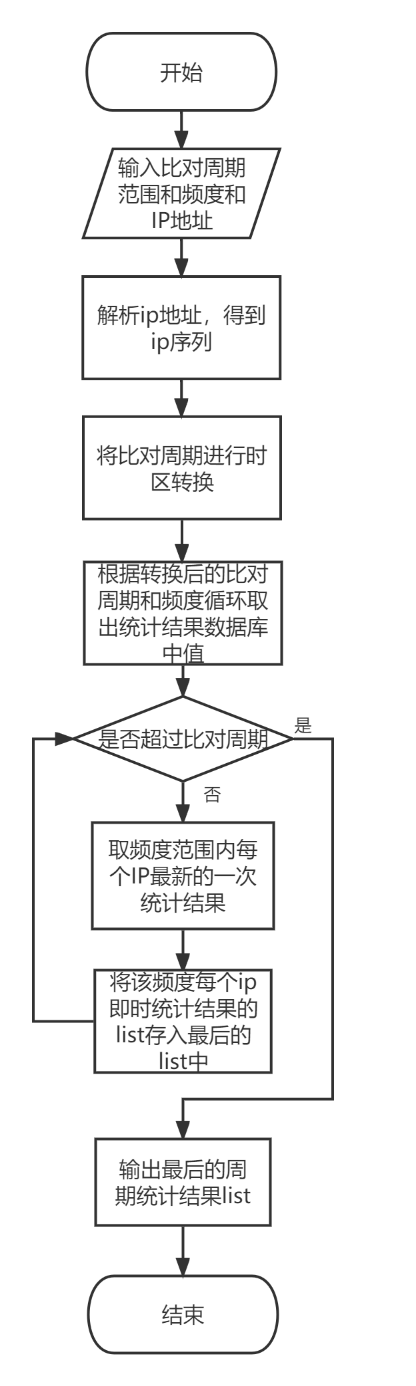


图2-13 ip读取周期性报告全机生成流程

上述IP读取的流程中，输入输出的参数格式如下表2-4所示。

表2-4 ip读取周期性报告参数列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 字段 | | | 备注 |
| 输入 | hostIp | | 单ip | 单个ip地址 |
| 多ip | 多个ip地址序列 |
| ip地址块 | 如192.168.132.0/24 |
| startTime | | | 周期开始时间，如  2022-07-14 00:00:00 |
| endTime | | | 周期结束时间，如  2022-07-22 00:00:00 |
| utc | | | 标记时区，如+8 |
| timeFrequence | | | 时间频度，如year month day hour 等 |
| multiple | | | 倍数，如1表示一个频度，如一天、一小时 |
| 输出 | ResultList | | | 每个周期内最新检查时间的即时报告信息，具体格式如下 |
| List*<*hostResult*>* HostList | | | 检查机器列表 |
| hostResult | checkTime | | 最新检查时间 |
| hostIp | | 检查的机器ip |
| failedList | | 失败的检查项列表（存放ruleID） |
| passList | | 成功的检查项列表（存放ruleID） |

#### 2.2.4 时区问题

不同机器可能存在不同的时区，而compliance service服务器也可能存在与不同的时区，所以为了解决时区不同的问题，这里提出以下解决方案：

（1）采集机器信息时，所采用的采集时间包括机器时间和机器时区，其中机器时间采用标准的utc 0时区的时间，然后在时区中存储，对应的utc偏移量，如2022-08-10 17:20:00，表示采集时utc零时区的时间， UTC +8，表示该机器所在的地区utc偏移量。

（2）比对时，存储在数据库中的时间，采用的机器的采集时间，所以数据库中对于不同机器，存储的均为零时区的时间。

（3）compliance Service获取报告时，可以输入一个时区信息，然后根据时区信息，将输入的时间转换成零时区的时间，然后在查询数据库即可，如compliance Service机器所在的时区为+8，则在查询数据库是，将每一段范围的开始，结束时间均-8，然后在查询数据库，此时查询出来的数据就是当前时区对应的一天。

## 3.数据库设计

### 3.1 ES数据库设计

#### 3.1.1 Debain10结果设计

当beat查询到信息后，将查询到的结果发送到es里存储，其存储项目如下表所示：

表3-1 es结果格式

|  |  |
| --- | --- |
| 表项 | 详细信息 |
| ID | 配置文件ID，标记使用的是哪个配置文件 |
| collectTime | 时间戳，标记检查时间（标准国际时间） |
| UTC | 时区偏移量，标记时区 |
| ruleId | 检查项ID，标记检查项 |
| result | 检查项结果，用于保存命令行运行的结果 |
| hostIp | 标记一台主机（终端ID） |

#### 3.1.2 Windows10结果设计

当beat查询到信息后，将查询到的结果发送到es里存储，其存储项目如下表所示：

表3-2 es结果格式

|  |  |
| --- | --- |
| 表项 | 详细信息 |
| ID | 配置文件ID，标记使用哪个配置文件 |
| collectTime | 时间戳，标记检查时间（标准国际时间） |
| UTC | 时区偏移量，标记时区 |
| ruleId | 检查项ID，标记检查项 |
| result | 检查项结果，用于保存命令行运行的结果 |
| hostIP | 标记一台主机 |

### 3.2 CS数据库设计

#### 3.2.1 配置文件表设计

本项目使用的数据库为postgreSQL数据库，由于时间的唯一性，所以本项目将存储的时间作为主键，为方便读取，可以设置一个ID项，从1开始，每整加一个版本加一，取值是只要根据ID取值即可，同时为确定到每一个检查项，所以需要ruleId项也作为主键，其数据库表结构下表3-3所示。

对于其中的type字段，表示处理采集结果的几种方式，type值的含义分别如下：

0：默认处理，即将采集到的字段和预期进行直接比对，判断比对值；

1：采集到的结果为access类型值，需要对结果值进行字符串处理，提取出其中的关键字段，然后再进行比对；

2：多文件处理，有些检查项涉及到多个文件，所以需要对其进行特殊比对；

3：不小于比对，即预期结果为不小于某个值；

4：不大于比对，即预期结果为不大于某个值；

表3-3 配置文件表

|  |  |  |
| --- | --- | --- |
| 表项 | 类型 | 备注 |
| cid | bigint | 主键，自增 |
| timestamp | timestamp | 时间戳，标记存储的时间 |
| id | integer | 配置文件ID，方便读取 |
| os | text | 标记分发的系统 |
| rule\_id | text | 检查项ID |
| data | text[] | 检查预期值 |
| type | integer | 比对类型 |
| param | text[] | 字符数组类型，存储多个命令行或path |

#### 3.2.2 比对结果表设计

查询的结果的比对，其结果值将会被存入数据库中，其数据库的表格式如下表3-4所示。

表3-4 检查报告表

|  |  |  |
| --- | --- | --- |
| 表项 | 类型 | 备注 |
| rid | bigint | 主键，自增 |
| timestamp | timestamp | 时间戳，标记采集的时间（统一），存储标准国际时间 |
| UTC | text | 标记UTC偏移量，如+2 |
| id | integer | 配置文件ID，方便读取 |
| os | text | 标记系统类型 |
| sever\_ip | inet | 标记负责比对的主机 |
| host\_ip | inet | 标记被比对的主机 |
| rule\_id | text | 标记是哪个检查项 |
| expected | text[] | 查询结果的预期值 |
| actual | text[] | 查询结果的实际值 |
| status | text | 比对的结果（pass/failed） |

#### 3.2.3 统计结果表设计

比对结果的统计数据，根据检查时间和机器ip的统计该机器在本次检查中，失败的检查项和成功的检查项，其表结果如下表3-5所示。

表3-5统计结果表

|  |  |  |
| --- | --- | --- |
| 表项 | 类型 | 备注 |
| sid | bigint | 主键，自增 |
| timestamp | timestamp | 机器采集时间，采用标准国际时间 |
| UTC | text | 标记UTC偏移量，如+2 |
| hostIp | inet | 检查的机器ip |
| failedList | text[] | 失败的检查项列表（存放ruleID） |
| passList | text[] | 成功的检查项列表（存放ruleID） |

## 3. 数据流图

数据流总体可以分为三大模块，分别是Web Service端、处理系统信息的Compliance Service模块以及存储系统信息的Elastic Search模块，具体如图3-1所示。

自顶向下的数据流流程是：

（1）直接操作数据库，将配置文件存储在数据库中，实现配置文件的下发操作；

（2）Collector模块从数据库中读取数据，并向LogAgent下发Windows10和Debain10的基线配置文件；

（3）simbeat读取系统信息并将检查结果存储在Elastic Search的数据库中。

自底向上的数据流流程是：

（1）Compliance Service的基线对比模块从Elastic Search的数据库获取检查项结果；

（2）Compliance Service模块从自身的数据库中获取检查项对比数据；

（3）Compliance Service对比检查项后生成检查报告，并存储到自身的数据库中；

（4）Web Service端可以从Compliance Service模块的数据库中获取检查报告。

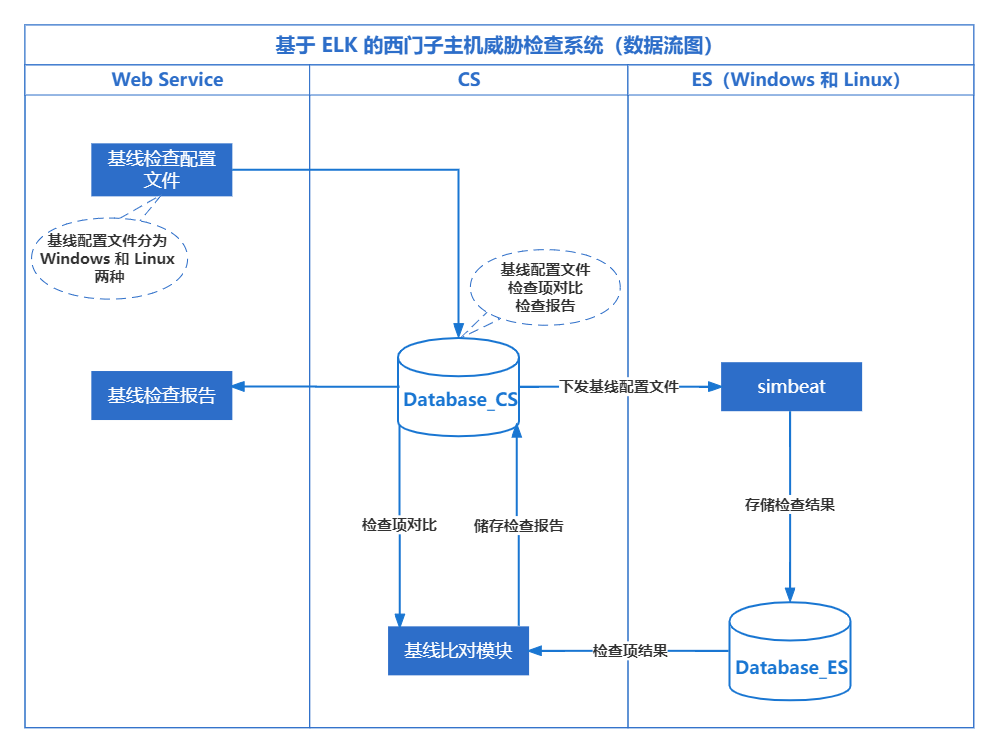


图3-1 数据流图