# 接口部分

## 2025.7.8接口文档

告警信息列表 + 修复前后的可视化对比图，都需要使用 **WebSocket**，允许客户端和服务端双向通讯。

以下是我整理的WebSocket前端大致流程

<https://obsius.site/143k320n541v0j5b2b2y>

**由于我们选择使用WebSocket协议进行前后端交互，Apifox没有办法在ws协议时添加响应参数，所以我在这对接口文档中的内容加以说明**

#### 获取告警列表信息

当我点击异常告警展示该view时，此时客户端需要得到服务器传来的告警列表，实时进行传输，客户端接收到则实时对列表刷新

ws://192.168.31.57:9000/getDangerousList

{  
 "listCount": [  
 {   
 "level": 0,  
 "count": 20  
 },  
 {  
 "level": 1,  
 "count": 32  
 },  
 {  
 "level": 2,  
 "count": 20  
 },  
 ],  
 "listData": [  
 {  
 "id": 0,  
 "status": "safe",  
 "time": "2025/07/11 11:32",  
 "dangerItem": {  
 "title": "CPU",  
 "content": "CPUCPUCPUCPUCPUCPUCPCUCPU我不想写了我累了我想放假休息",  
 "dangerData": 100000  
 },  
 },  
 {  
 "id": 1,  
 "status": "danger1",  
 "time": "2025/07/11 11:32",  
 "dangerItem": {  
 "title": "disk",  
 "content": "diskdiskdiskdiskdiskdiskdiskdisk",  
 "dangerData": 2000  
 },  
 },  
 {  
 "id": 2,  
 "status": "danger2",  
 "time": "2025/07/11 11:32",  
 "dangerItem": {  
 "title": "diskio",  
 "content": "diskiodiskiodiskiodiskiodiskiodiskio",  
 "dangerData": 999  
 },  
 }  
 ],  
}

* listCount：记录告警总数

* 1. level：分为三种类型，0：重要告警；1：告警总数（重要+普通）；2：已处理告警 **int** **必填**

* 1. count：数量 **integer** **必填**

* listData：告警条信息

* 1. id：唯一标识 **integer** **必填**

* 1. status：告警状态 **String** **必填**

* + 1. ”safe“：已处理告警

* + 1. ”danger1“： 重要告警

* + 1. ”danger2“： 普通告警

* 1. dangerItem：告警具体信息

* + 1. title：告警主题，例如CPU全占满 **String** **必填**

* + 1. content：告警详细内容 **String** **必填**

* + 1. dangerData：告警值 **String** **必填**

#### 获取告警可视化图表

当用户在客户端点击单个告警条，显示该告警详细信息以及处理结果，该接口旨在发送该条告警的报警数据，例如某个数据超越某阈值，展示该报警数据产生环境，所以主要交互数据为可视化图表类型和可视化图表数据

ws://192.168.31.57:9000/getDangerousData

{  
 "id": 0  
}

* id：告警唯一标识 **integer** **必填**

{  
 "id": 0,  
 "type": "line01",  
 "data": {  
 "timedata": [  
 "00:00",  
 "01:15",  
 "02:30",  
 "03:45",  
 "05:00",  
 "06:15",  
 "07:30",  
 "08:45",  
 "10:00",  
 "11:15",  
 "12:30",  
 "13:45",  
 "15:00",  
 "16:15",  
 "17:30",  
 "18:45",  
 "20:00",  
 "21:15",  
 "22:30",  
 "23:45"  
 ],  
 "valuedata": [  
 300, 280, 250, 260, 270, 300, 550, 500, 400, 390, 380, 390, 400, 500, 600, 750, 800, 700, 600, 400  
 ],  
 "dangerdata": [  
 [  
 {  
 "name": "告警名称",  
 "begin": "07:30"  
 },  
 {  
 "end": "10:00"  
 }  
 ],  
 [  
 {  
 "name": "告警名称",  
 "begin": "17:30"  
 },  
 {  
 "end": "21:15"  
 }  
 ]  
 ]  
 }  
}

* id：告警唯一标识 **integer** **必填**

* type：展示可视化图表类型 **String** **必填**

* data：可视化图表数据 **父节点**

* 1. timedata：时间节点，按时间序列标准，单位看具体数据 **Array** **必填**

* 1. valuedata：数据值，每个节点上获取到的数据大小 **Array 必填**

* 1. dangerdata：存告警具体区域 **Array** **必填**

* + 1. name：告警名称 **String** **必填**

* + 1. begin：超出阈值开始时间 **String** **必填**

* + 1. end：超出阈值结束时间 **String** **必填**

{  
 "id": 1,  
 "type": "line02",  
 "data": {  
 "valuedata": [  
 ["00:30",117],  
 ["01:00",130],  
 ["01:30",86],  
 ["02:00",42],  
 ["02:30",200],  
 ["03:00",180],  
 ["03:30",178],  
 ["04:00",130],  
 ["04:30",150],  
 ["05:00",190],  
 ["05:30",198],  
 ["06:00",140],  
 ["06:30",150],  
 ["07:00",110],  
 ["07:30",120],  
 ["08:00",158],  
 ["08:30",176],  
 ["09:00",302],  
 ["09:30",189],  
 ["10:00",199],  
 ["10:30",210],  
 ["11:00",260],  
 ["11:30",290],  
 ["12:00",280],  
 ["12:30",222]  
 ],  
 "thresholddata": 300  
 }  
}

* id：告警唯一标识 **integer** **必填**

* type：展示可视化图表类型 **String** **必填**

* data：可视化图表数据 **父节点**

* 1. valuedata：数据值，内容分别为时间节点和该节点上的数据 **Arrray** **必填**

* 1. thresholddata：阈值数据 **integer** **必填**

#### 获取已修复告警可视化图表

当用户在客户端点击告警信息时，需要显示已修复后的数据以及修复日记，该接口用于对该类数据进行传输。可视化部分内容与【获取告警可视化图表】接口类似，但需添加一类修复日记

ws://192.168.31.57:9000/getSafeData

{  
 id: 0  
}

* id：告警唯一标识 **integer** **必填**

{  
 "id": 0,  
 "type": "line01",  
 "worknote": [  
 {  
 "step": "步骤一",  
 "content": [  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"  
 ]  
 },  
 {  
 "step": "步骤二",  
 "content": [  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"  
 ]  
 },  
 {  
 "step": "步骤三",  
 "content": [  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"  
 ]  
 },  
 // ... 依次类推  
 ],  
 "data": {  
 "timedata": [  
 "00:00",  
 "01:15",  
 "02:30",  
 "03:45",  
 "05:00",  
 "06:15",  
 "07:30",  
 "08:45",  
 "10:00",  
 "11:15",  
 "12:30",  
 "13:45",  
 "15:00",  
 "16:15",  
 "17:30",  
 "18:45",  
 "20:00",  
 "21:15",  
 "22:30",  
 "23:45",  
 // 为凸显对比，需要对时间节点进行延长  
 "01:00",  
 "02:15",  
 "03:30",  
 "04:45",  
 "06:00",  
 "07:15",  
 // ... 依次类推  
 ],  
 "valuedata": [  
 300, 280, 250, 260, 270, 300, 550, 500, 400, 390, 380, 390, 400, 500, 600, 750, 800, 700, 600, 400,  
 // 时间节点延长，数据节点也需要相应  
 320, 222, 350, 333, 310, 222 // ... 依次类推  
 ],  
 // 告警信息不需要处理，此时为已处理结束的告警内容  
 "dangerdata": [  
 [  
 {  
 "name": "告警名称",  
 "begin": "07:30"  
 },  
 {  
 "end": "10:00"  
 }  
 ],  
 [  
 {  
 "name": "告警名称",  
 "begin": "17:30"  
 },  
 {  
 "end": "21:15"  
 }  
 ]  
 ]  
 }  
}

* id：告警唯一标识 **integer** **必填**

* type：展示可视化图表类型 **String** **必填**

* worknote：展示修复日记内容 **Array** **必填**

* 1. step：步骤标题，比如步骤一：我修复了balabala **String** **必填**

* 1. content：具体修复内容 **Array 必填**

* data：可视化图表数据 **父节点**

* 1. timedata：时间节点，按时间序列标准，单位看具体数据 **Array** **必填**

* 1. valuedata：数据值，每个节点上获取到的数据大小 **Array 必填**

* 1. dangerdata：存告警具体区域 **Array** **必填**

* + 1. name：告警名称 **String** **必填**

* + 1. begin：超出阈值开始时间 **String** **必填**

* + 1. end：超出阈值结束时间 **String** **必填**

{  
 "id": 1,  
 "type": "line02",  
 "worknote": [  
 {  
 "step": "步骤一",  
 "content": [  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"  
 ]  
 },  
 {  
 "step": "步骤二",  
 "content": [  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"  
 ]  
 },  
 {  
 "step": "步骤三",  
 "content": [  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",  
 "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"  
 ]  
 },  
 // ... 依次类推  
 ],   
 "data": {  
 "valuedata": [  
 ["00:30",117],  
 ["01:00",130],  
 ["01:30",86],  
 ["02:00",42],  
 ["02:30",200],  
 ["03:00",180],  
 ["03:30",178],  
 ["04:00",130],  
 ["04:30",150],  
 ["05:00",190],  
 ["05:30",198],  
 ["06:00",140],  
 ["06:30",150],  
 ["07:00",110],  
 ["07:30",120],  
 ["08:00",158],  
 ["08:30",176],  
 ["09:00",302],  
 ["09:30",189],  
 ["10:00",199],  
 ["10:30",210],  
 ["11:00",260],  
 ["11:30",290],  
 ["12:00",280],  
 ["12:30",222], //  
 ["13:00",200],  
 ["13:30",178],  
 ["14:00",130],  
 ["14:30",150],  
 ["15:00",190],  
 ["15:30",198],  
 ["16:00",140],  
 ["16:30",150],  
 ["17:00",110],  
 ["17:30",120],  
 ["18:00",140],  
 ["18:30",150],  
 ["19:00",110],  
 ["19:30",120],  
 ["20:00",100],  
 ["20:30",201],  
 ["21:00",187]  
 ],  
 "thresholddata": 300  
 }  
}

* id：告警唯一标识 **integer** **必填**

* type：展示可视化图表类型 **String** **必填**

* worknote：展示修复日记内容 **Array** **必填**

* 1. step：步骤标题，比如步骤一：我修复了balabala **String** **必填**

* 1. content：具体修复内容 **Array 必填**

* data：可视化图表数据 **父节点**

* 1. valuedata：数据值，内容分别为时间节点和该节点上的数据 **Arrray** **必填**

* 1. thresholddata：阈值数据 **integer** **必填**

#### 获取全盘扫描数据

该接口用于获取全盘扫描数据，后端实时发送正在扫描以及扫描列表和漏洞信息。【正在扫描】数据在扫描不同指标时，下一个指标会将上一个指标的内容覆盖；【扫描列表】和【漏洞信息】则是根据扫描进度添加

{  
 // 当此时我扫描到【disk】时，该数据覆盖为"disk"  
 "title": "cpu",  
 "scanlist": [  
 {  
 "id": 0,  
 "name": "cpu在干嘛",  
 "content": "cpu你在干嘛呀",  
 },  
 {  
 "id": 1,  
 "name": "disk在干嘛",  
 "content": "disk你在干嘛呀",  
 },  
 {  
 "id": 2,  
 "name": "diskio在干嘛",  
 "content": "diskio你在干嘛呀"  
 }  
 ],  
 "looplist": [  
 {  
 "id": 0,  
 "name": "漏洞名称",  
 "content": "漏洞内容"  
 },  
 {  
 "id": 1,  
 "name": "第二个漏洞",  
 "content": "第二个漏洞内容是什么呀"  
 },  
 {  
 "id": 2,  
 "name": "怎么还有漏洞？",  
 "content": "你这漏洞有点多吧，快点改，告诉我漏洞内容是什么！！"  
 }  
 ]  
}

* title：扫描到的部分，比如此时我扫描到cpu，则展示cpu，扫描到了disk，则覆盖为disk。只允许覆盖不允许增加或删除 **String** **必填**

* scanlist：扫描列表 **父节点**

* 1. id：扫描列表的每一条的id，唯一标识 **integer** **必填**

* 1. name：每扫描到一个功能点或者“需要重点注意的点”时，记录该部分名称 **String** **必填**

* 1. content：同上，记录该部分详细信息 **String** **必填**

* looplist：漏洞列表 **父节点**

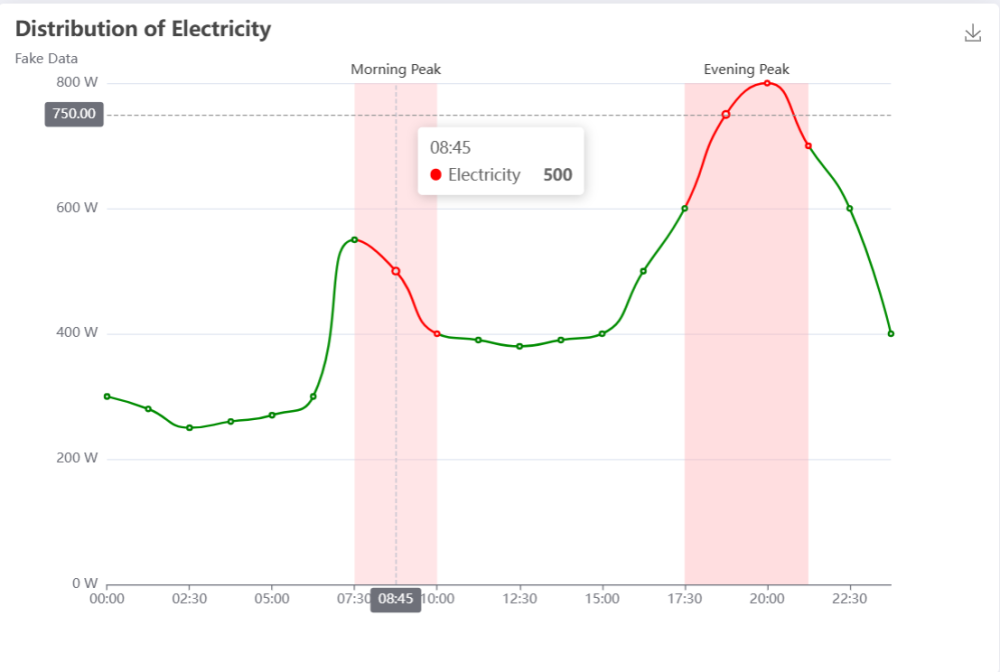
* 1. id：扫描到的漏洞信息每一条id，唯一标识 **integer** **必填**

* 1. name：扫描到的漏洞名称 **String** **必填**

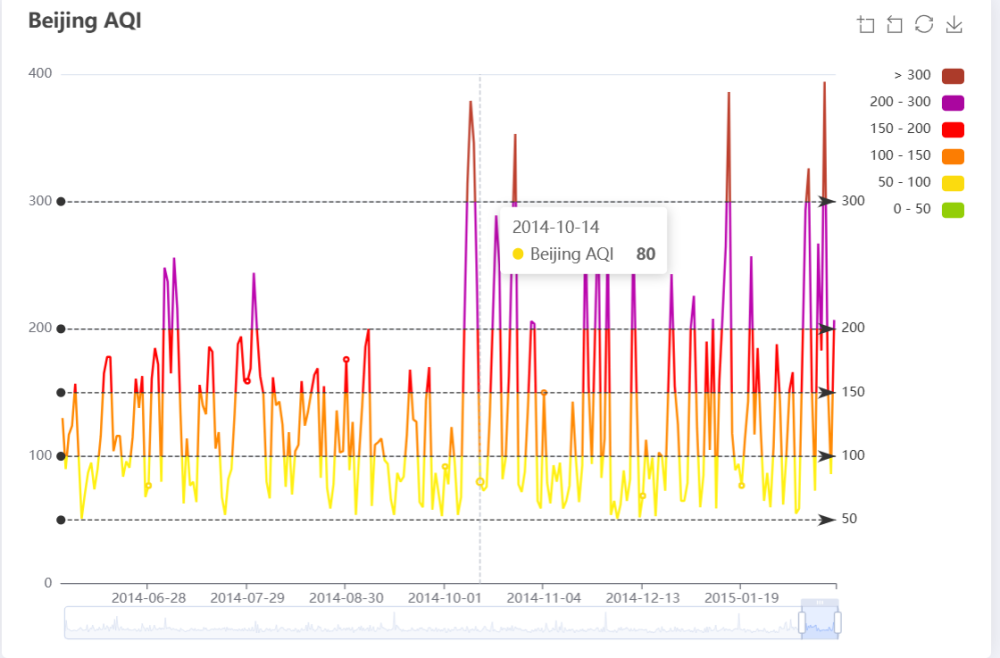
* 1. content：扫描到的漏洞详细信息 **String** **必填**

### 数据图选择

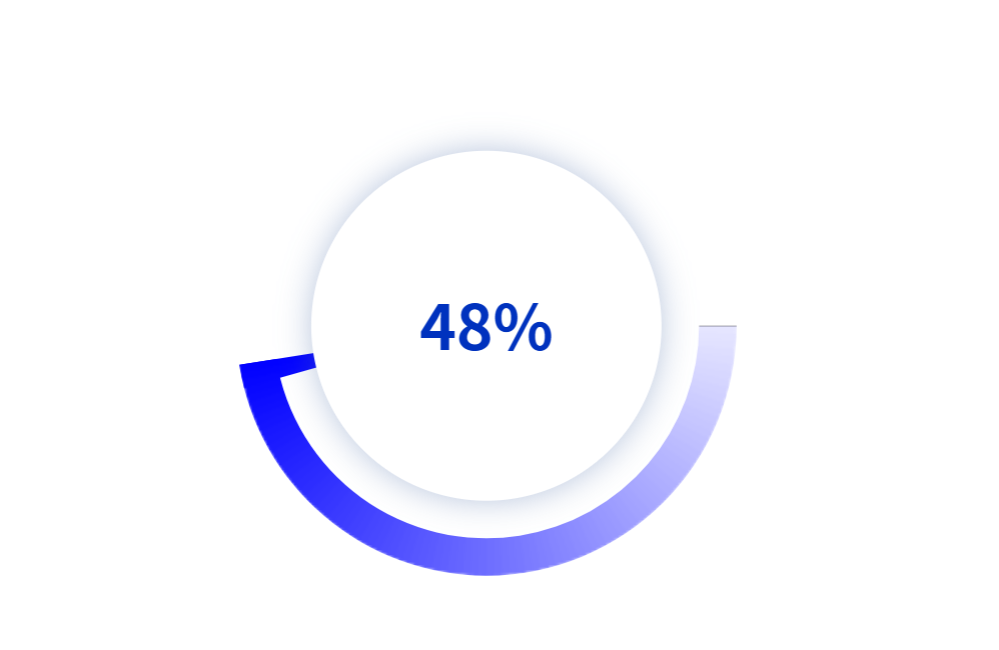
<https://echarts.apache.org/examples/zh/editor.html?c=line-sections>



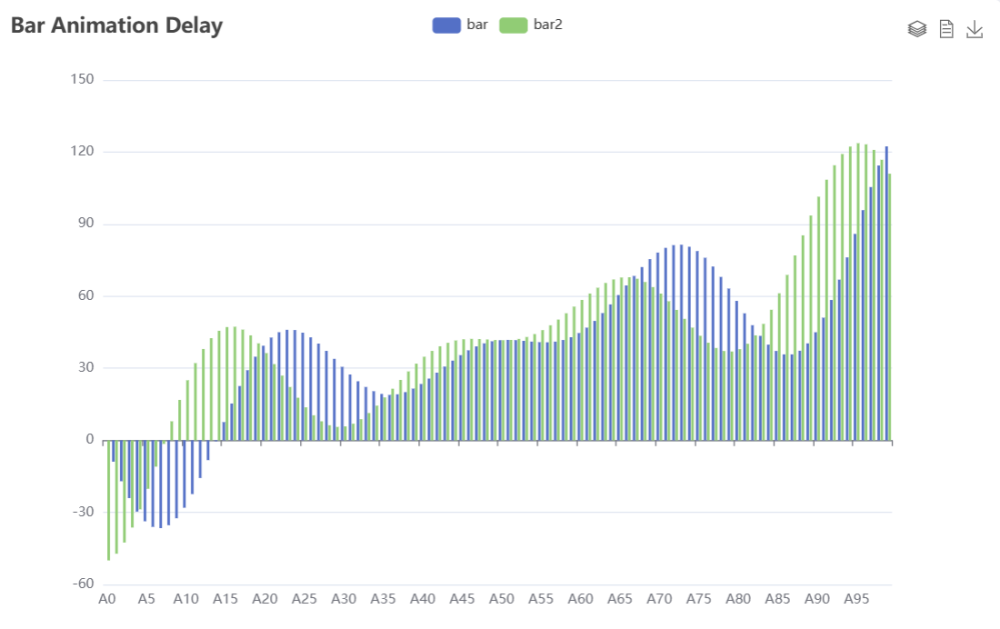
<https://echarts.apache.org/examples/zh/editor.html?c=line-aqi>



<https://echarts.apache.org/examples/zh/editor.html?c=custom-gauge>



<https://echarts.apache.org/examples/zh/editor.html?c=bar-animation-delay>



<https://echarts.apache.org/examples/zh/editor.html?c=candlestick-sh>



## 总结02

### 对于【获取告警可视化】这部分接口

思路如下

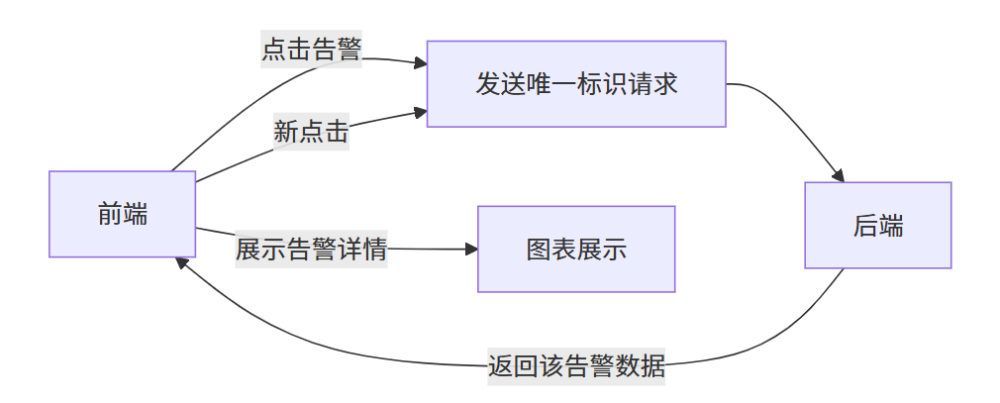
**第一种传参方式：**

对多条告警展示时，点击进入某条告警条查看详细内容时，这个时候ws向后端发送唯一标识请求，后端拿到告警id并对处理返回告警数据，数据内容如接口响应数据所示，前端获取之后对图表进行展示。当退出当前告警详情时，仍然允许连接持续，到下一次点击传参时向后端推送更新。断开连接持续到我退出告警列表时

* **保持WebSocket连接**：仅通过同一连接发送新的ID参数

* **后端实时响应新数据**：通过send()推送更新

// 前端示例（切换ID时）  
function changeChart(newId) {  
 // 复用现有连接发送新ID  
 websocket.send(JSON.stringify({ id: newId }));  
}



**第二种传参方式**：

建立连接之后持续推送所有的告警数据，前端使用内存存储，在每次推送告警数据的时候更新内存存储，在客户端点击告警条时前端筛选展示对应数据。

但是该方法对前端数据处理压力大，且数据传输量大（考虑到基本信息还有可视化数据内容），内存占用高则导致数据推送不及时。所以选择第一种即可。

