选拔赛设计文档

[作品名称]
设计文档
（根据作品匿名要求，在所有作品资料中请勿出现学校、团队以及队员身份等相关信息）
所在赛道与赛项：*A-ST*

一、目标问题与意义价值

说明作品的应用领域，解决或关注的问题，实现的目标与基本功能，以及作品的理论意义或应用价值。

随着大数据技术和人工智能的迅速发展，数据的创建量呈指数级增长，数据的应用场景也愈加广泛。而保证数据安全的采集、传输、储存机制尚不明确，从而个人隐私泄露事件日益频发。而数据安全是数字经济发展的压舱石，是国家安全的重要组成部分，数据安全已成为事关国家安全和经济社会发展的重大议题。
现有面向个人用户的检测工具（例如Firefox monitor）依据多样但不明确，且检测的数据部分来自存储着已泄露数据的数据库，并不能反映个人隐私信息的泄漏情况的问题。而企业级别的数据安全与隐私解决方案不突出关注个人隐私数据的泄露情况，并且网页是个人隐私的高频泄露途径，因此有必要构建基于网页内容进行个人信息泄露风险评估工具。
为了能够对数据内容的潜在隐私风险做出评估，设计了个人隐私信息泄露风险评估系统，针对多源数据进行个人信息层面的泄露风险评估，用于即使发现数据隐患、保证数据安全，在此前提下充分发掘数据要素价值及推动生产力发展。系统收集指定目标的多源数据，可以根据不同的数据隐私需要，匹配不同数据项，基于《中华人民共和国数据安全法》和《中华人民共和国个人信息保护法》的隐私信息定义进行泄露风险的确定，相较于其他检测方案更加符合国家法律法规，并可以根据用户需求调整不同关联数据项的风险等级，满足不同源数据存在的差异化风险需求。

二、设计思路与方案

阐述作品解决问题的主要设计思路与技术路线，以及详细的设计方案。
解决了什么问题，使用了什么技术以及技术，阐述总体方案。

设计思路

建立一套个人信息泄露主动发现系统。该系统能够针对特定重点监控目标主动爬取所有该目标发布的数据，通过数据解析技术还原出原始数据内容，并结合机器学习和正则表达式技术，监测是否存在信息泄露情况。系统为用户提供监测目标、检测模型规则等配置管理功能，以及已监测过结果查询功能。具体要实现特定目标网站的数据爬取与解析模块；信息泄露规则检测模块；基于WEB的系统配置管理模块；系统配置模块。

技术路线

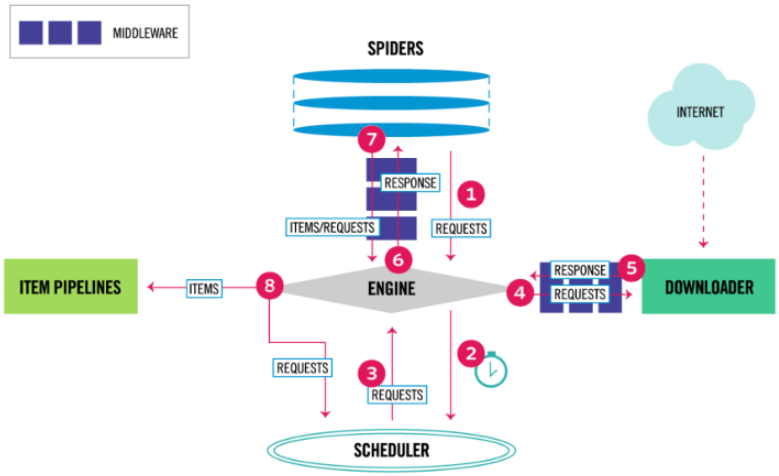
2.1基于scrapy和redis的分布式爬虫技术

问题描述

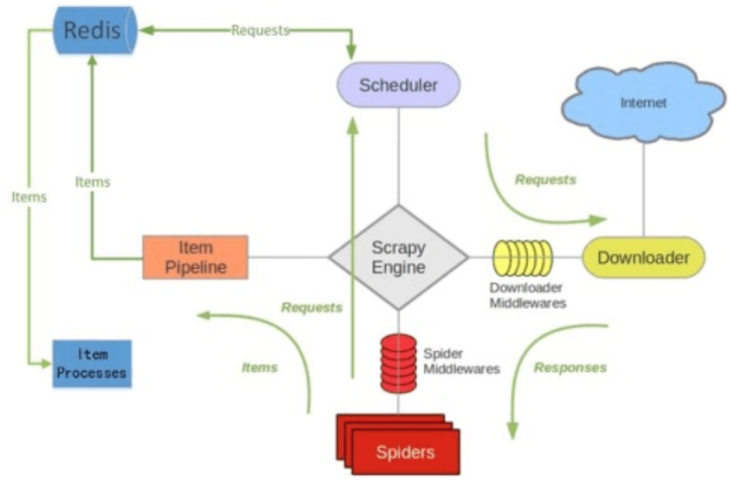
普通的爬虫一般由一个进程进行所有的数据爬取，如果程序意外中断，就需要重新启动爬虫，不能恢复已获得数据，并且极大的影响到任务执行。分布式的数据获取可以将爬虫部署在多个地址下，多个爬虫单独运行并共享同一个爬取队列，一个爬虫程序的异常不会影响到任务的整体执行结果，并且可以动态调整系统中的爬虫数量，以获得更好的系统运行效率。

解决方案

Scrapy是适用于Python的一个快速、高层次的屏幕抓取和web抓取框架，用于抓取web站点并从页面中提取结构化的数据，是一个通用的爬虫框架，其结构如图所示。



Scrapy-redis是为了更方便地实现Scrapy分布式爬取，而提供了一些以redis为基础的组件(仅有组件)。关键组件如下：
Scheduler：Scrapy多个spider不能共享待爬取队列Scrapy queue， 即Scrapy本身不支持爬虫分布式，scrapy-redis 的解决是把这个Scrapy queue换成redis数据库（也是指redis队列），从同一个redis-server存放要爬取的request，便能让多个spider去同一个数据库里读取。
Duplication Filter：它通过redis的set不重复的特性，实现了Duplication Filter去重。scrapy-redis调度器从引擎接受request，将request的指纹存redis的set检查是否重复，并将不重复的request push写redis的 request queue。
scrapy-redis的总体思路：这套组件通过重写scheduler和spider类，实现了调度、spider启动和redis的交互。实现新的dupefilter和queue类，达到了判重和调度容器和redis的交互，因为每个主机上的爬虫进程都访问同一个redis数据库，所以调度和判重都统一进行统一管理，达到了分布式爬虫的目的。
当spider被初始化时，同时会初始化一个对应的scheduler对象，这个调度器对象通过读取settings，配置好自己的调度容器queue和判重工具dupefilter。
每当一个spider产出一个request的时候，scrapy引擎会把这个reuqest递交给这个spider对应的scheduler对象进行调度，scheduler对象通过访问redis对request进行判重，如果不重复就把他添加进redis中的调度器队列里。当调度条件满足时，scheduler对象就从redis的调度器队列中取出一个request发送给spider，让其进行爬取。
当spider爬取的所有暂时可用url之后，scheduler发现这个spider对应的redis的调度器队列空了，于是触发信号spider_idle，spider收到这个信号之后，直接连接redis读取strart_url池，获取新的一批url入口，然后再次重复上面的工作，如图所示。



在实际程序运行中，打开爬虫，并向队列中加入起始URL，爬虫获取对应URL下内容，并获得新的URL加入待爬取队列，直到没有新的URL加入。

2.2基于scrapy信号机制的任务流程控制技术

问题描述

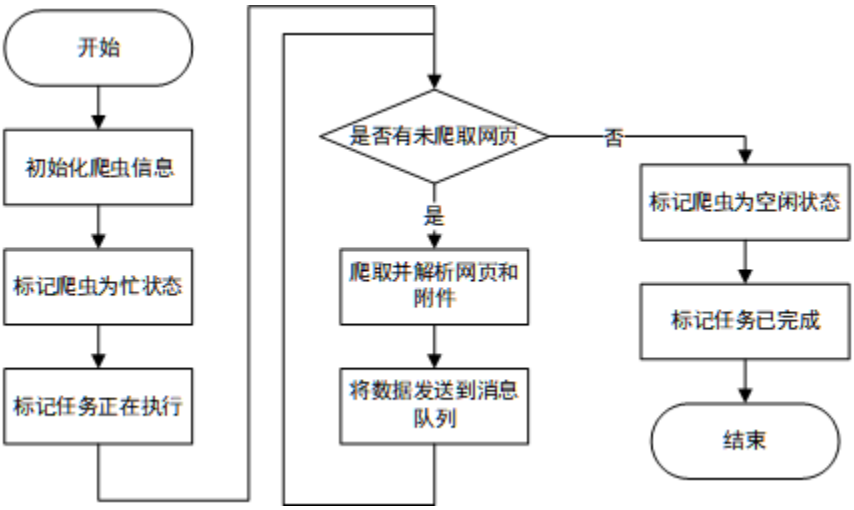
在数据获取程序执行过程中，需要根据爬虫执行过程改变任务执行记录的对应字段，并且检测任务的执行情况，需要利用到scrapy的组件以及信号机制，并且配合后台进程对执行中任务进行监测。

解决方案

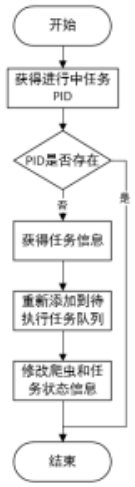
利用pipeline组件实现爬取内容的发送，即时将网页内容发送给消息队列。利用MIddleWare以及信号机制进行程序执行关键点（开始执行、获取到内容、执行结束等）的跟踪，及时修改任务记录信息。scrapy的项目管道(Pipeline)：负责处理爬虫从网页中抽取的实体，主要的功能是持久化实体、验证实体的有效性、清除不需要的信息。当页面被爬虫解析后，将被发送到项目管道，并经过几个特定的次序处理数据。主要有个三个方法：

process_item(self,item,spider):实现对item数据的处理
open_spider(self, spider)：在爬虫开启的时候仅执行一次
close_spider(self, spider)：在爬虫关闭的时候仅执行一次

Scrapy广泛使用信号来通知特定事件发生的时间。 可以在Scrapy项目中捕获一些这些信号（例如，使用扩展名）来执行其他任务或扩展Scrapy的功能。支持的信号包含爬虫初始化、返回数据、触发异常、爬虫关闭、爬虫空闲、发起下载请求、请求到达引擎等，程序中利用信号机制实现了程序爬取网页的速度和平均延迟的平衡，程序的异常检测，有利于提高程序的稳定性。运行过程如图所示。



此外，系统还需要控制程序对数据获取进程进行监控，并恢复中断的进程。如图



2.3基于rabbitMQ的分布式系统架构

问题描述

系统中的爬虫将网页上或者网页附件中爬取到的数据，利用Scrapy信号机制在Middleware组件中封装好item数据项，传递给检测模型。过程涉及到如何传递给检测模型的问题。

解决方案

RabbitMQ是实现了高级消息队列协议（AMQP）的开源消息代理软件（亦称面向消息的中间件），它是消费者（consumer）-生产者（producer）模型的一个典型代表，具备高效可靠的消息异步传递机制,其工作就是接收和转发消息。producer往消息队列中不断写入消息，consumer则可以读取或者订阅队列中的消息。在传递信息的过程中具备可靠的安全机制来保证信息传递过程中的安全性。MQ相当于一个中介，生产方通过MQ与消费方交互，它将应用程序之间进行了解耦，在分布式系统中应用十分广泛。其基本结构如图1所示。

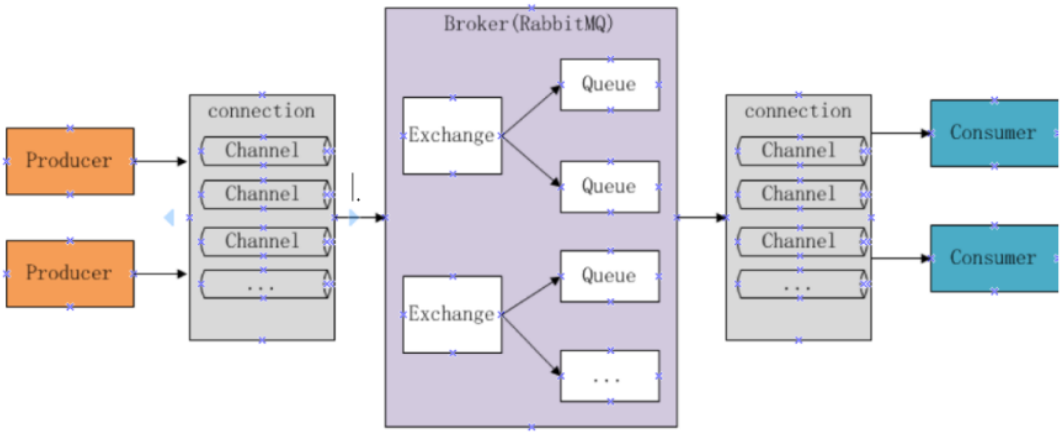


图1
如图2所示，Rabbitmq系统最核心的组件是交换机（Exchange）和队列（Queue）。Exchange类似于数据通信网络中的交换机，提供消息路由策略。producer不是通过信道直接将消息发送给queue，而是先发送给Exchange。一个Exchange可以和多个Queue进行绑定，producer在传递消息的时候，会传递一个绑定建（ROUTING_KEY），Exchange会根据这个ROUTING_KEY按照特定的路由算法，将消息路由给指定的queue。

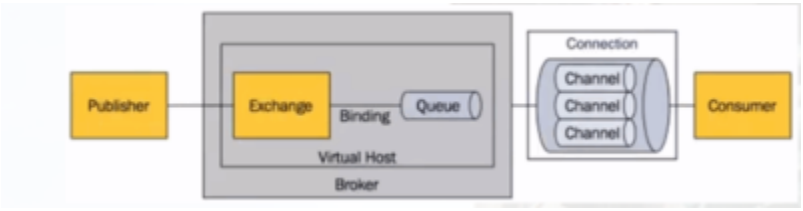


图2

如图3所示，rabbitmq中提供了多种可选择的交换机类：直连交换机（direct），主题交换机（topic），（头交换机）headers和 扇型交换机（fanout）。因系统中存在不同的检测模型，故数据将根据需要传递给不同的检测模型。而直连交换机将会对绑定键（binding key）和路由键（routing key）进行精确匹配，从而确定消息该分发到哪个队列。故在本系统中采用直连交换机。

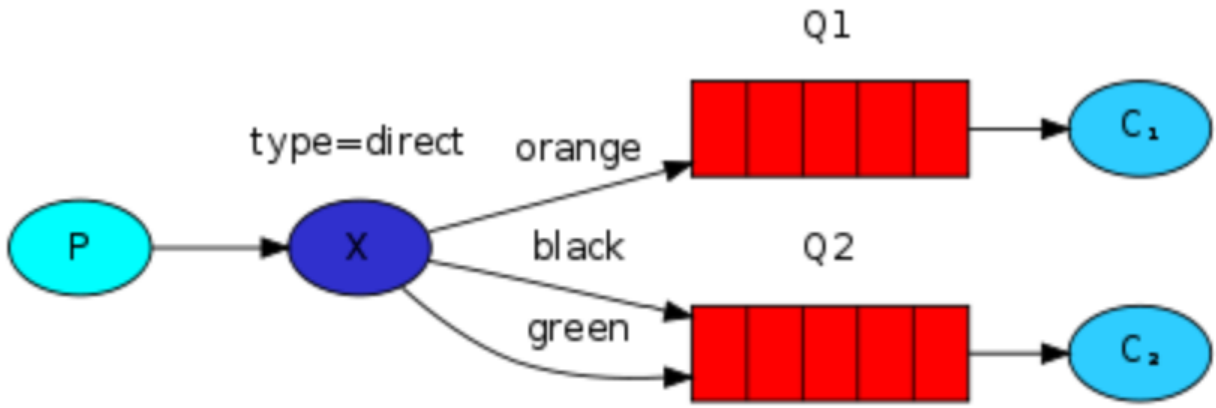


图3

如上图所示，我们可以看到直连交换机 X和两个队列进行了绑定。第一个队列使用orange作为绑定键，第二个队列有两个绑定，一个使用black作为绑定键，另外一个使用green。这样以来，当路由键为orange的消息发布到交换机，就会被路由到队列Q1。路由键为black或者green的消息就会路由到Q2。其他的所有消息都将会被丢弃。

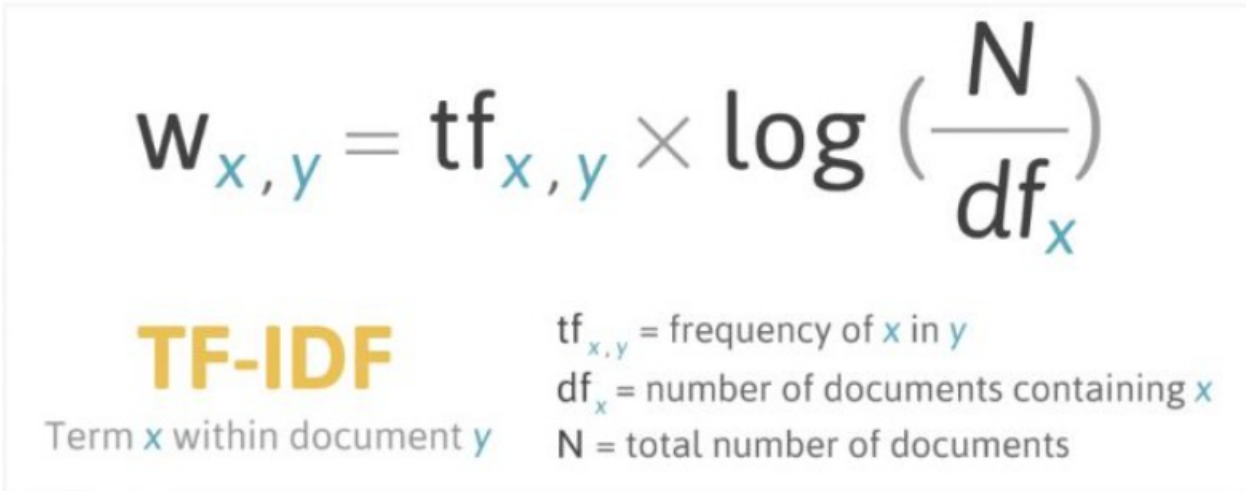
2.4基于正则匹配和机器学习的泄露风险项提取技术

问题描述

个人隐私泄露主动发现系统需要主动爬取待检测的网页，之后要对爬取到的内容进行隐私检测，而判断是否构成个人隐私信息泄露不单需要获取到数据，还要对获取到的数据进行处理，从中提取涉及到个人信息的数据。

解决方案

在Scrapy的Spider组件将网页文本或文件解析并封装为一个数据项之后，利用Scrapy的信号机制在Middleware组件中实现正则匹配模块，对下载信息进行匹配。如果是网页文本内容，模块对网页文本按照任务指定的泄露规则对应的正则表达式进行扫描；如果是文件内容，模块通过文件类型和文件路径对文件内容进行提取，将文件内容提取为一个字符串，再用正则表达式进行匹配检测，并记录匹配到的内容，对应的规则名称，检测时间，来源网页和任务名称。正则表达式的匹配需要用到python中的re库。而机器学习部分，先对爬取到的部分数据按空格或逗号句号进行分割，然后通过jieba分词库对每句进行文本分割，对处理之后的文本开始用TF-IDF算法进行单词权值的计算，然后去掉停用词，通过贝叶斯预测种类。

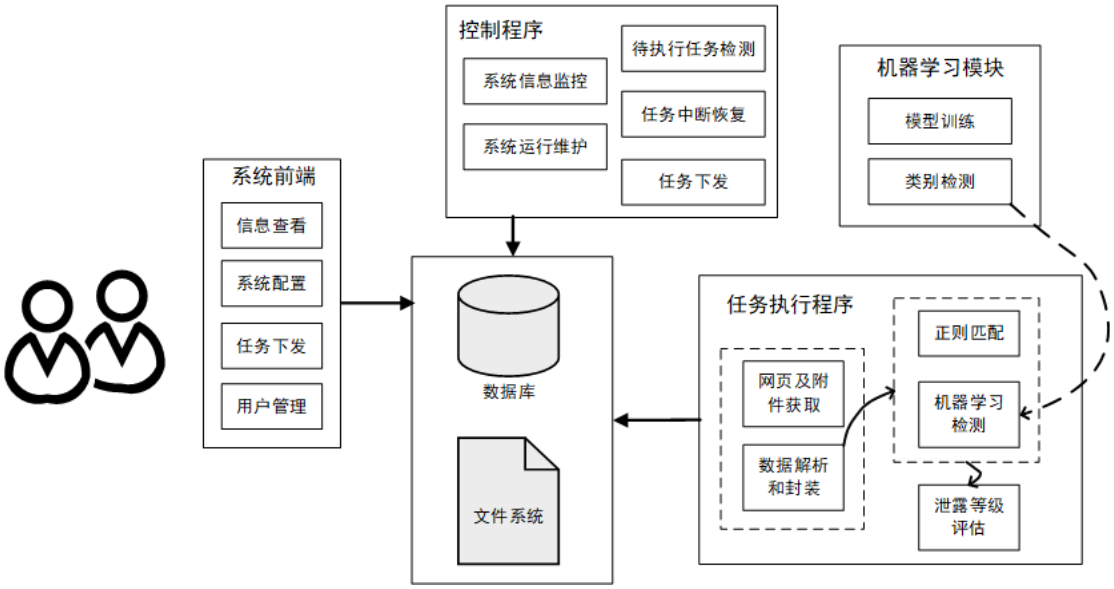


TF-IDF算法

实现框架

系统基于B/S框架，由系统前端、后台控制程序、任务执行程序、机器学习模块以及相应的数据存储，包括数据库和文件系统共同组成。

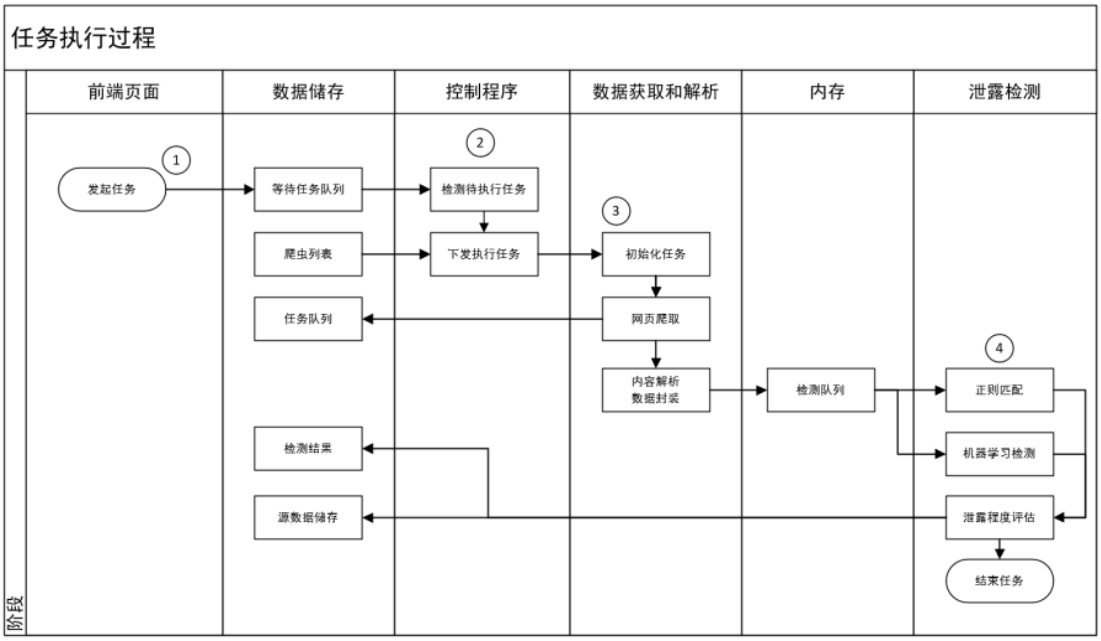
系统的**前端**部分只起到记录用户操作和向用户展示信息的功能。对用户操作的处理逻辑都由系统的**控制程序**完成，后台控制程序是独立于系统前端的管理程序，控制程序负责检测系统中是否有待执行任务并持续更新和维护系统信息。如果控制程序检测到待执行任务，就会调用任务执行程序完成任务。**任务执行程序**的工作是完成系统的扫描任务，并将扫描结果保存到数据库，任务执行分为**数据爬取**和**泄露检测**两部分，泄漏检测需要进行数据信息的归类和是否构成写泄露的判断，数据爬取由爬虫完成，在爬虫获取到网页数据后，分别由**正则匹配**和**机器学习检测**敏感信息内容以及类别，再由泄露判断部分完成对是否构成泄露的分析。机器学习模型分为训练和检测两部分，训练部分用于模型检测自身的构建，由人工利用数据集进行训练，不对系统检测程序可见，检测程序只调用机器学习模块的检测功能，用于判断符合的敏感信息种类。系统的实现框架如图所示。



系统实现框架

系统的核心功能是完成一次检测任务，网页监控管理功能可以视为周期性完成同一件扫描任务，而临时扫描任务则是仅完成一次扫描，系统辅助测试则是完成一次简单的测试扫描，不同任务的区别只在任务添加方式和使用爬虫上有区别，在执行过程上都相同。

以临时任务为例说明系统任务的执行过程，如图



任务执行流程图

（1）临时任务由用户在Web网页上下发，储存在等待任务队列中。

（2）控制程序检测到有待执行任务，会根据任务类型选择可用爬虫，如果有可用爬虫，就会分配爬虫来获取任务需要的数据，进行任务的下发；如果没有可用爬虫，会保持任务的阻塞。

（3）在收到任务执行调用后，爬虫会首先根据任务参数进行初始化，改变任务执行状态和自身状态（标记任务正在进行和自身忙状态），随后进行网页数据的获取，在爬取并检测完成全部的页面和文件后，任务会结束。

（4）在获得爬虫封装的数据后，泄漏检测模块会分两步进行泄露情况的检测：首先同时利用正则匹配和机器学习对文本中的存在敏感信息进行查找的归类，获得检测内容中的敏感信息及其对应类别；然后根据已检测出的信息类别进行判断，评估出结合已得到敏感信息类别，可构成的泄露风险等级。

数据获取模块和泄漏检测模块共同构成执行任务程序，泄漏检测程序是数据获取程序生命周期的延续，可以通过数据获取的信号完成整体程序结束的判断。

三、方案实现

说明作品实施方案（如具体的软、硬件技术及集成方法），整体所达成的具体功能或服务。

详细设计

系统网络结构

系统运行于服务器上，为用户提供网页作为系统外部接口，用户需要拥有可访问服务器的网络设备方可使用系统，通过浏览器访问系统提供的页面，用户可使用系统功能。因爬虫需对网页进行扫描，所以服务器主机需要连接到互联网。 理想的系统部署如图1所示。

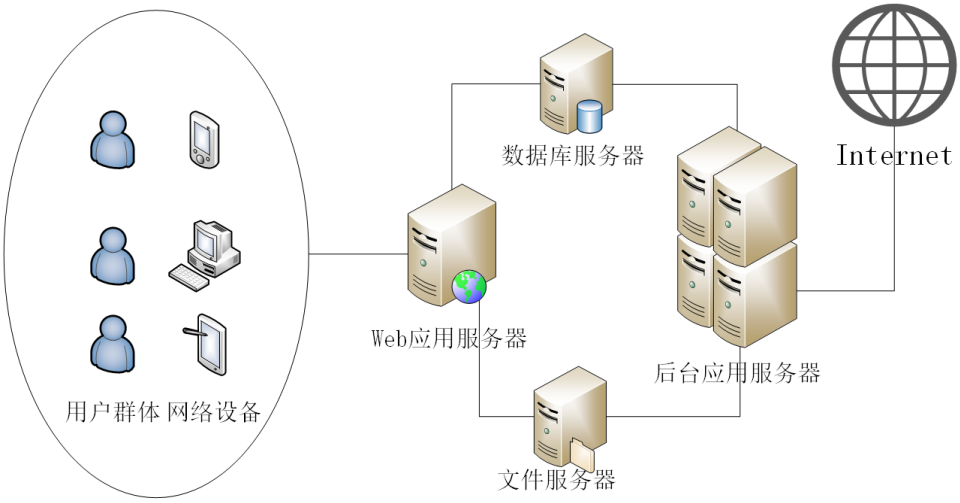


图3.1

再最小化部署中，提供网页的前台程序和负责业务逻辑的后台程序，以及系统的数据库、文件存储系统部署在同一台服务器主机上。如图 2所示

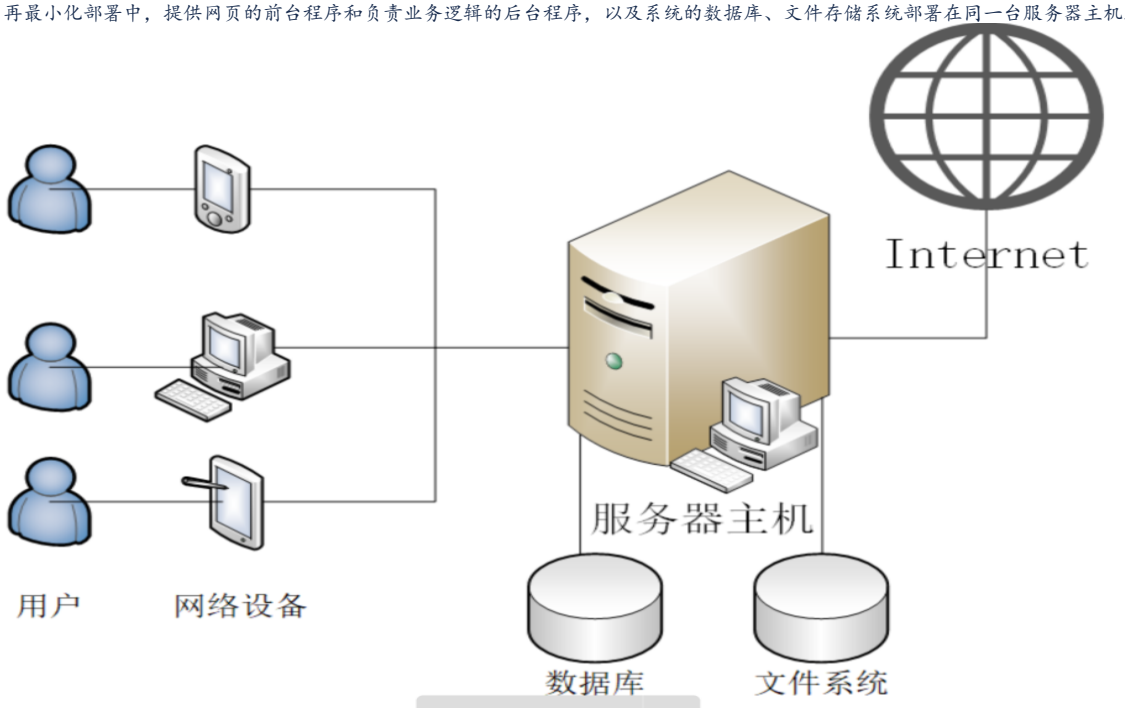


图3.2

方案设计

3.1总体框架设计

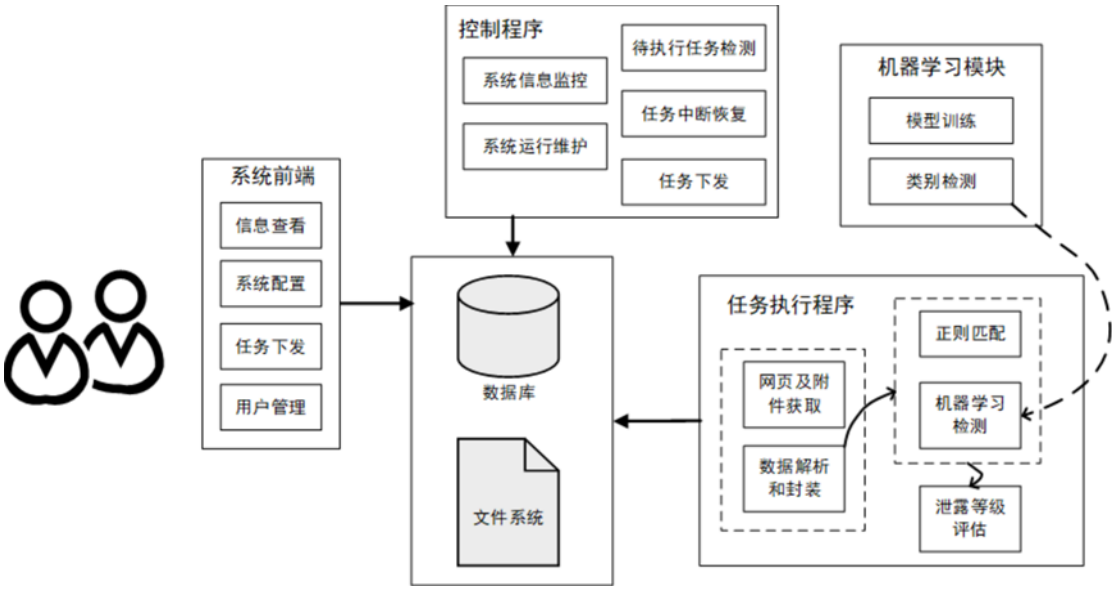


图3.3

系统总体由前端模块、控制模块、任务执行模块、机器学习模块组成，对各模块的描述如下表所示：

模块划分	模块描述
前端模块	提供用户界面和系统对应用户功能的外部接口
控制模块	调度任务和系统信息维护
任务执行模块	爬取网页和对内容进行正则匹配，并将结果数据保存到数据库和文件
机器学习模块	对机器学习模型进行训练，并为任务执行模块提供机器学习检测

3.2前端模块设计

前端模块为用户提供查看系统状态、添加临时任务、配置监测规则、配置周期任务等一系列系统功能的前端页面，使用sqlachemy和数据库交互，记录用户操作和下发的临时任务，并将任务信息记录到文件系统中。前端模块组成图如下：

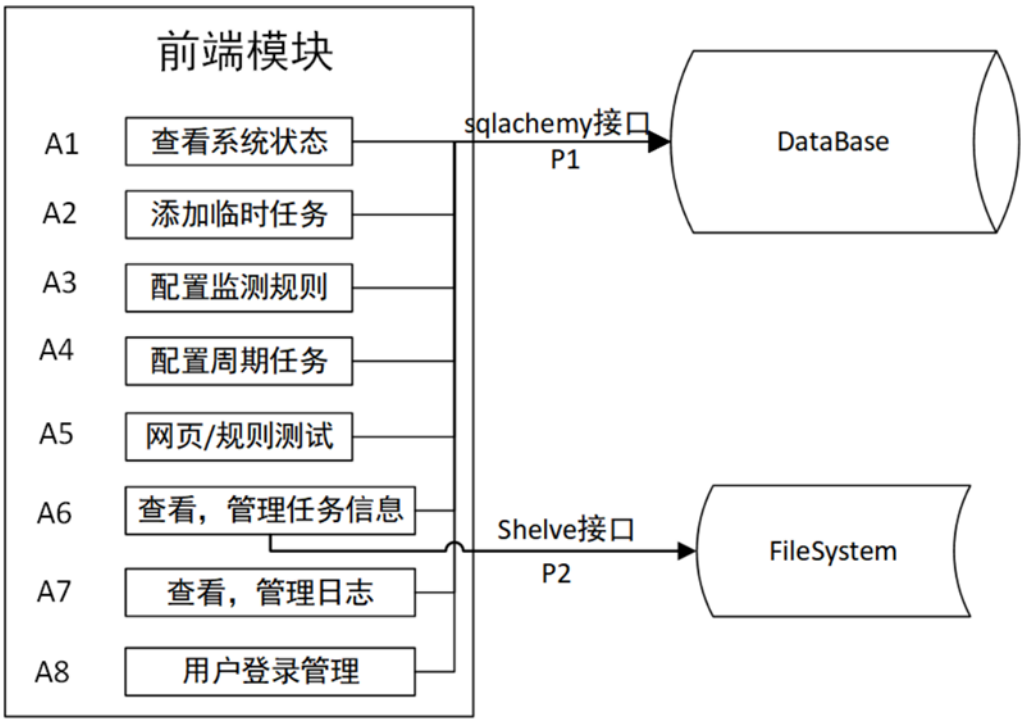


图3.4

前端模块中各子模块的功能如下：

编号	子模块名称	子模块功能
A1	查看系统状态	在登录首页用户可以查看系统任务数量和机器资源相关数据
A2	添加临时任务	用户发起临时检测任务的入口，将任务信息写入数据库
A3	配置监测规则	用户进行查看，删除，增加正则规则的操作，将结果保存到数据库
A4	配置周期任务	用户查看，修改，增加，删除周期任务配置，将结果保存到数据库
A5	网页/规则测试	用户进行网页，规则的测试，将测试任务信息写入数据库
A6	查看、管理任务信息	用户查看任务记录、泄露数据
A7	查看、管理日志	用户查看系统日志和用户日志，数据来自数据库日志表
A8	用户登录管理	根据用户表进行登录验证，利用session维持和清除登录信息

3.3控制模块设计

控制模块负责调度任务和系统信息维护，是系统管理控制层面和核心，在后台常驻，脱离于前端模块独立运行，可以在关闭了系统网页后继续处理未完成任务，一定程度上保证了系统的稳定运行。其中初始化运行信息、系统运行信息维护以及检测任务都是通过sqlachemy存入数据库。控制模块的组成如图所示。

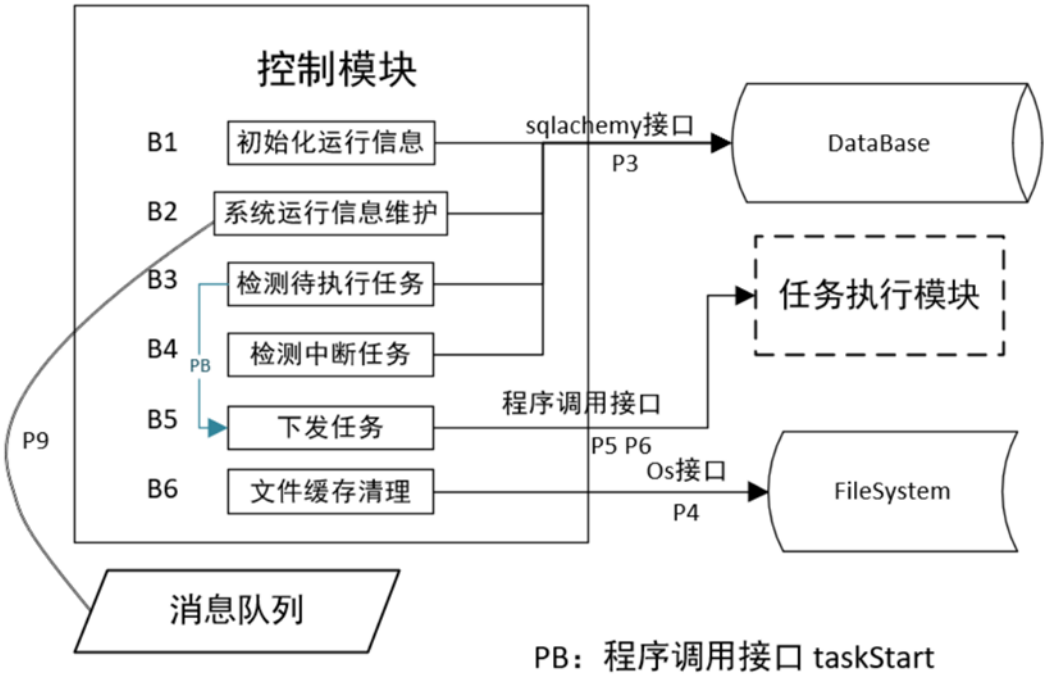


图3.5

控制模块中各子模块的功能如下：

编号	子模块名称	子模块功能
B1	初始化运行信息	读取周期任务配置表，安排周期任务，实例化程序所需数据结构
B2	系统运行信息维护	清理过期的周期任务爬取记录，更新资源使用情况
B3	检测待执行任务	读取待执行任务表，若有任务，下发任务
B4	检测中断任务	查看正在执行任务的pid是否存在，不存在将任务写入待执行任务表
B5	下发任务	执行空闲Scrapy任务爬虫，进行任务
B6	文件缓存清理	定期删除系统的文件缓存

3.4任务执行模块

任务执行模块负责任务执行，被控制模块调用时才会运行，，由数据获取和解析封装以及信息匹配和泄露评估两部分组成。其中数据获取和解析封装由初始化爬虫、修改任务信息、网页信息下载、数据解析封装这四个子模块组成，信息匹配和泄露评估由正则匹配、机器学习检测、泄露风险评估三个子模块组成。如图所示。

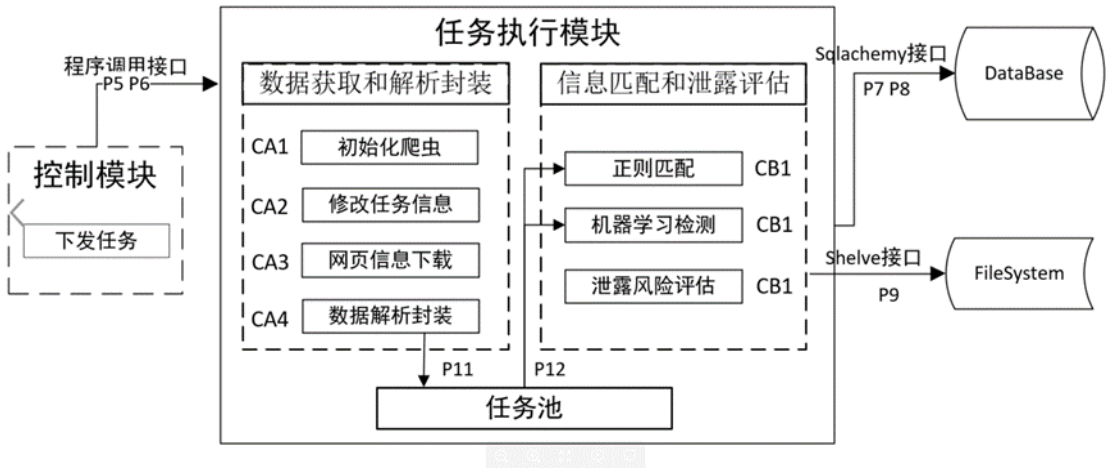


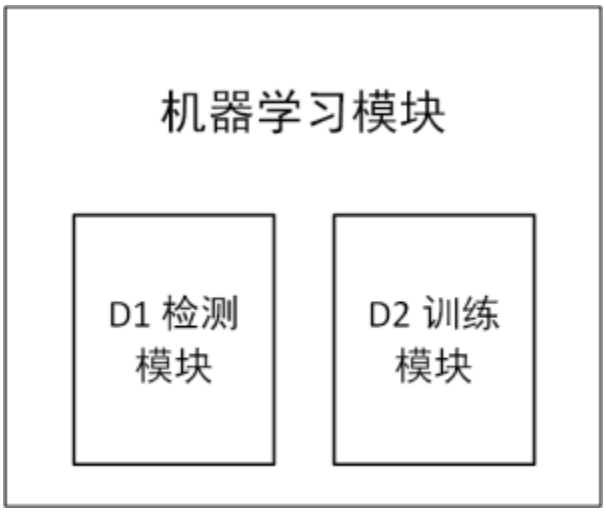
图3.6

任务执行模块中各子模块的功能如下：

编号	模块名称	模块功能
CA1	初始化爬虫	根据传入的参数，初始化爬虫启动需要的信息
CA2	修改任务调度信息	将待执行任务表中任务转移到执行任务表中，修改爬虫状态
CA3	网页信息下载	解析网页Html标签中的文本，下载网页附件
CA4	数据解析封装	将文件解析为文本内容，封装为数据项，网页文本则直接封装
CB1	正则匹配	利用正则规则匹配文本，储存匹配到的数据
CB2	机器学习检测	利用机器学习模型匹配文本，储存匹配到的数据
CB3	泄露风险评估	根据匹配到的数据判断该网页信息泄露风险等级，并写入结果到数

3.5机器学习模块

机器学习模块负责训练机器学习模型，其由检测模块和训练模块两个模块组成，与任务执行模块部分复用机器学习检测模块，其组成如图3.9所示。



编号	模块名称	模块功能
D1	检测模块	对训练中的模型进行阶段检测，比较模型准确度以判断是否停止训练。并提供信息泄露检测的接口。
D2	训练模块	根据输入的训练集信息对模型进行训练。

接口设计

3.1、内部接口
系统的内部接口包含各模块之间的联系，以及模块和系统数据存储之间的联系。系统中模块间的接口为控制模块调用任务执行模块的程序调用接口，其余为系统模块与数据库之间的接口及系统与文件储存的接口。内部接口的类型的定义如表4.2所示，接口的编号见功能模块设计图。系统涉及的接口有九个，分为五种，分别是数据库接口，文件系统读写接口文件删除接口，程序调用接口和创建新进程接口，各接口负责不同功能，连接相应模块。
表4.2 系统内部接口表

接口编号	接口类型	
P1	sqlachemy接口	前端模块-数据库
P2	Shelve接口	前端模块-文件系统
P3	sqlachemy接口	控制模块-数据库
P4	Os接口	控制模块-文件系统
P5	程序调用接口	控制模块-数据获取模块
P6	程序调用接口	控制模块-泄漏检测模块
P7	sqlachemy接口	数据获取模块-数据库
P8	sqlachemy接口	泄漏检测模块-数据库
P9	Shelve接口	泄漏检测模块-文件系统
P10	程序调用接口	泄露检测模块-检测模块
P11	addItem接口	数据获取模块-消息队列
P12	detect接口	消息队列-泄漏检测模块
P13	通讯接口	消息队列-控制模块

P1前端模块-数据库：

接口名称	接口说明
LoginCheck	登录检查
GetInfo	获得系统信息
GetRunningTasks	获得运行中任务
GetPrdTaskCfgs	获得周期任务配置
GetRules	获得系统规则
LaunchTask	添加临时任务
GetAllData	获得所有检测数据
EditPrdCfg	修改周期任务配置
AddPrdCfg	添加周期配置
DelPrdCfg	删除周期配置
SwitchPrdCfg	开启关闭周期任务
DelPrdData	删除周期任务配置
GetPrdDataByName	根据任务名称获得周期任务数据
GetTempTask	获得临时任务列表
DelTempTask	删除临时任务及其数据

GetTaskDataByName	获得任务详情

P2前端模块-文件系统：前端页面需要查看用于数据检测的源数据，这部分数据由检测程序储存、系统操作模块定时清理。在需要查看时，服务器后台根据数据检测时间和url从文件系统中调取用于检测的文本，显示在前端页面上。

	接口名称（eg）	返回值	参数（type：name）	接口说明
1	GetText（detectTime, url）	字符串：获得到的文本 int 0：未找到数据	1.str：detectTime 数据检测时间2019-02-1823:40:00, 2.str：url 检测数据来源https://www.blogs.com/1	通过数据检测时间和网页url，提取到系统用于检测的文本，若未找到文本，返回0。

P3控制模块-数据库：数据获取模块将数据发送给消息队列，消息队列将数据传递给数据检测模块。

	接口名称	返回值	参数（type：name）	接口说明
1	sysInfo	int 1：更新成功 int 0：更新未成功	无	获取并更新系统信息
2	FindTask	Dict：（startUrl，taskName，domain，taskID,taskType）任务信息字典 0：无待执行任务 -1查询未成功	无	寻找是否有待执行任务
3	prdHistoryDel	int：num 删除条数 int：-1 删除未成功	int：interval 表示有效天数	删除周期时长以外的周期记录历史
4	FindErrTask	Dict：（startUrl，taskName，domain，taskType）任务信息字典 0：无中断行任务 -1查询未成功	无	寻找中断任务
5	AddTask (name，startUrl，domain，taskType)	int 1：插入成功 int 0：插入未成功	str：name str：startUrl str：domain str：taskType	添加一条任务到任务队列，被周期调用，实现周期任务功能
6	ReadPrdConfig	list[Dict]：[(name，startUrl，domain，rules，taskType),...]配置信息字典 -1查询未成功	无	读取系统的全部周期任务配置
7	FindModifiedPrd	list[Dict]：[(name，startUrl，domain，rules，taskType),...]配置信息字典 -1查询未成功	无	找到需要修改的周期任务配置

P4控制模块-文件系统：

	接口名称	返回值	参数（type：name）	接口说明
1	DelFile（timeOfDay）	int 1：删除成功 int 0：删除未成功	int：timeOfDay	删除指定天数之前的记录文件

P5控制模块-数据获取模块：控制模块将任务的起始地址发送给爬虫的爬起列表，爬虫在检测到待爬取网页后，自动开始数据获取。

	接口名称	返回值	参数（type：name）	接口说明
1	StartTask （startUrl，taskName，domain，taskID，taskType）	int 1：任务开始 int 0：任务未开始	str：startUrl str：taskName str：domain int：taskID str：taskType	将新的任务开始url加入到带爬取列表，爬虫对其自动进行爬取

P6控制模块-泄漏检测模块：

	接口名称	返回值	接口说明

1	StartTest（）	int 1：检测开始 int 0：检测未开始	开启检测模块，监听消息队列的待检测任

P7数据获取模块-数据库：

	接口名称	返回值	参数（type：name）	接口说明
1	AddHistory（url，time，taskName）	int 1：写入开始 int 0：写入未成功	str：url str：time str：taskName	将周期任务的检测记录写入到数据库
2	FindHistory（url，time，taskName）	int：1爬取过 int：0未爬取过	str：url str：timeOfDay str：taskName	查找某个任务的url在给定时间内有

P8泄漏检测模块-数据库：

	接口名称	返回值	参数（type：name）	接口说明
1	log （url，taskName,content，taskid,taskType,time）	int 1：写入开始 int 0：写入未成功	str：url str：taskName str：content int：taskID str：taskType int：time	将泄露检测到的内容记录到数据

P9泄漏检测模块-文件系统：

	接口名称	返回值	参数（type：name）	接口说明
1	shelve.open(filename，flag='c'，protocol=None，writeback=False)	int 1：打开/创建成功 int 0：打开/创建失败	str：filename str：flag int：protocol bool:writeback	将泄露检测过的数据记录到文件系统中

P10泄露检测模块-检测模块：

	接口名称	返回值	参数（type：name）	接口说明
1	mldetect （url，taskName,content，taskid,taskType,time）	int 1：检测成功 int 0：检测失败	str：url str：taskName str：content int：taskID str：taskType int：time	对消息队列或机器学习需要检测的

P11数据获取模块-消息队列：

	接口名称	返回值	参数（type：name）	接口说明
1	addItem	int 1：加入开始 int 0：加入失败	str ： item	将爬虫获取的item给

P12消息队列-泄漏检测模块：

	接口名称	返回值	参数（type：name）	接口说明
1	detect	int 1：检测成功 int 0：检测失败	str ： item	消息队里将item传

P13消息队列-控制模块：

	接口名称	返回值	接口说明
1	startRabbitmq	int 1：消息队列开启成功 int 0：消息队列开启失败	

3.2、外部接口

外部接口为系统页面提供给用户的操作页面，数据信息以及操作按钮。因为系统在页面设计上有较多共同点，存在较多的接口复用情况，所以在外部接口设计中对系统中重要的外部接口进行分类介绍，如下表

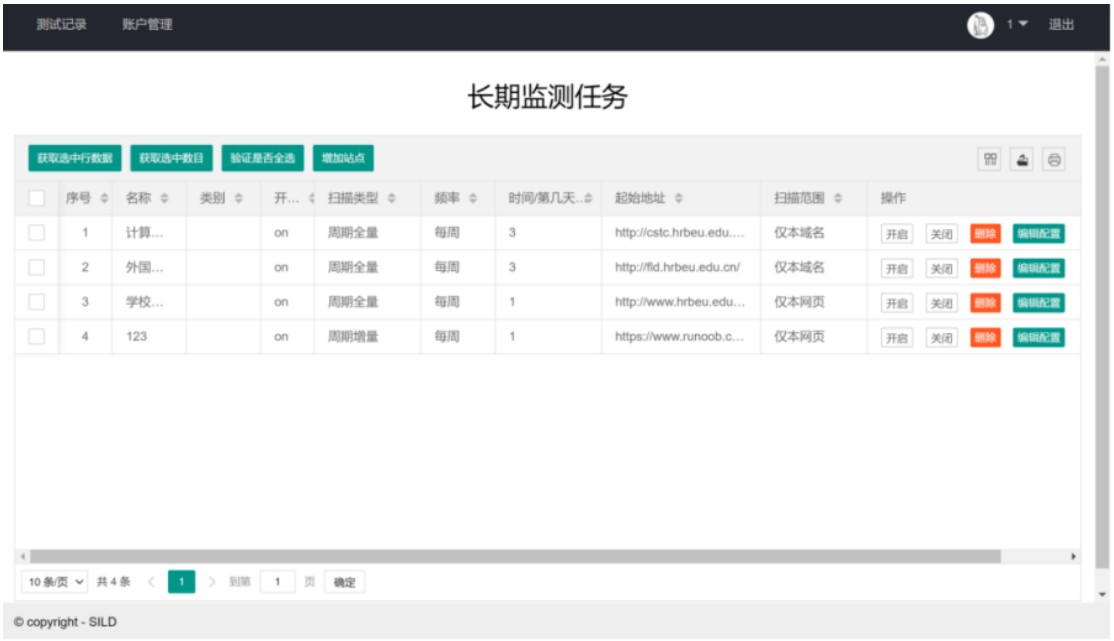
接口名称	接口功能
登录及退出按钮	登录按钮提供从系统登录页面进入系统主页的外部接口，退出登录按钮提供清除用户的登录信息并返回到系统登录页面的外部接口。
导航栏	导航栏包含左侧的页面导航栏和顶部的功能导航栏，能根据用户点击，跳转到系统提供的不同的页面。
表单提交及重置按钮	用户点击提交按钮提交表单后，系统会根据业务逻辑在数据库中存储用户操作，用户可以通过重置按钮来清除表单内容。
表格操作按钮	表格操作按钮包含了获取选中行数据、筛选数据列、导出数据到文件和打印数据等操作，用户通过这些按钮来对表格中的数据进行针对性的查看和导出功能。
表格数据搜索按钮	在泄露信息表格中，用户根据规则名称和内容进行泄露数据的模糊搜索。
表格数据项操作按钮	用户通过点击数据表每行后方的按钮来进行对数据项的操作。

四、运行结果/应用效果

说明作品（系统）实际工作/运行的情况或效果。
效果截图
系统主界面

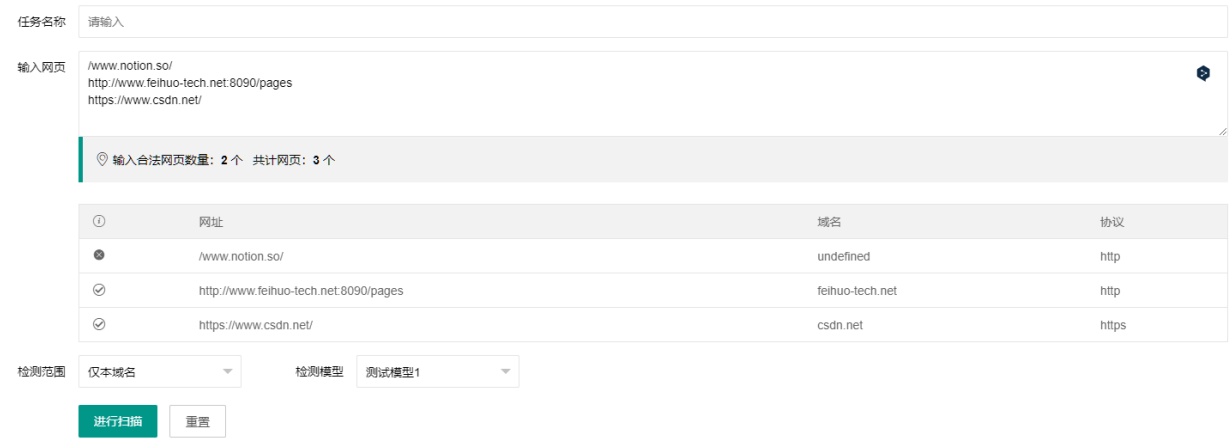
图1
如图1所示，系统主界面中系统信息一栏显示当前系统运行期间，磁盘和内存占用情况以及流量接受和发送情况。系统任务一栏中显示系统当前待执行任务以及执行中的任务，对于执行中的任务会显示当前正在执行的任务以及该任务的执行进度。

长期检测任务界面



长期监测任务界面可显示监测任务名称以及任务类型及频率。用户可以点击"增加站点"对自己想监测的目标进行周期性的监测。也可以修改相应的配置

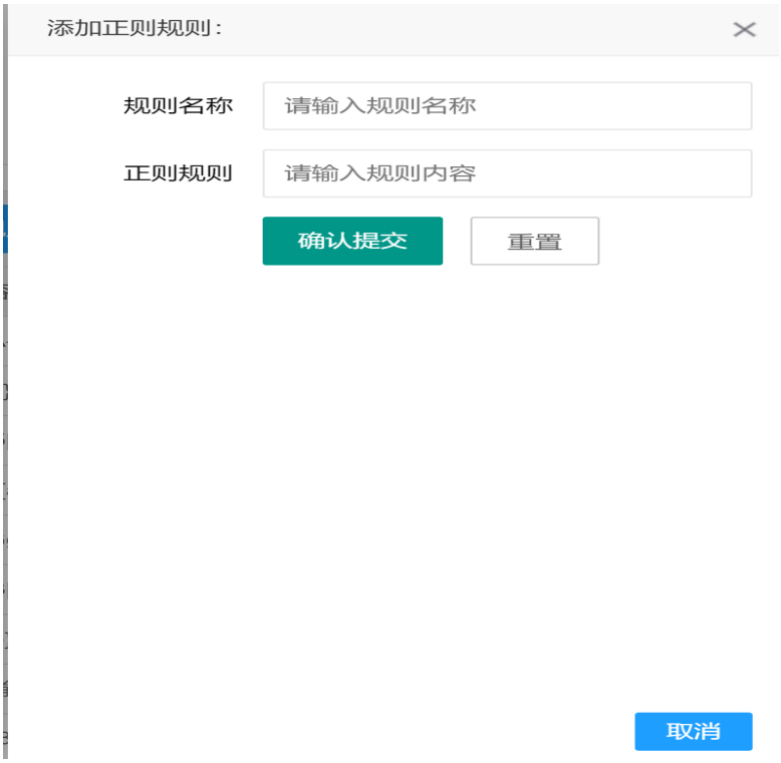
监测任务界面



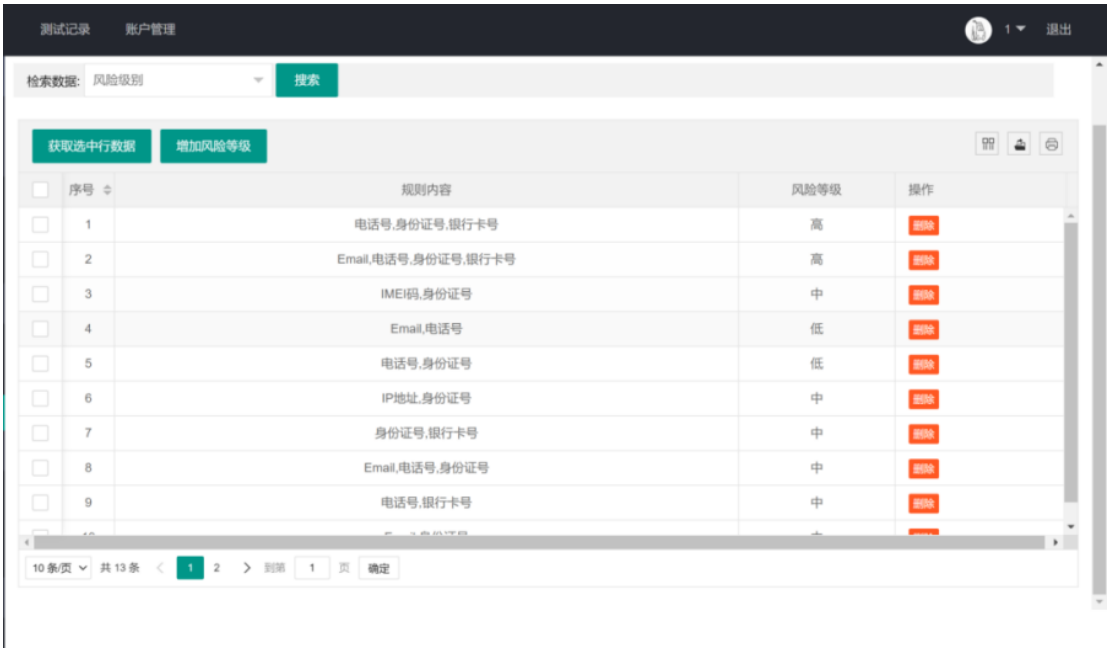
如图4所示，临时监测任务界面可以下发一个临时监测任务。用户输入任务名称以及想要监测的网页，选择检测范围以及检测输入网页的模型，该页面可以判断用户输入的网页是否合法，不合法的给与错误提示。

泄露检测模型页面

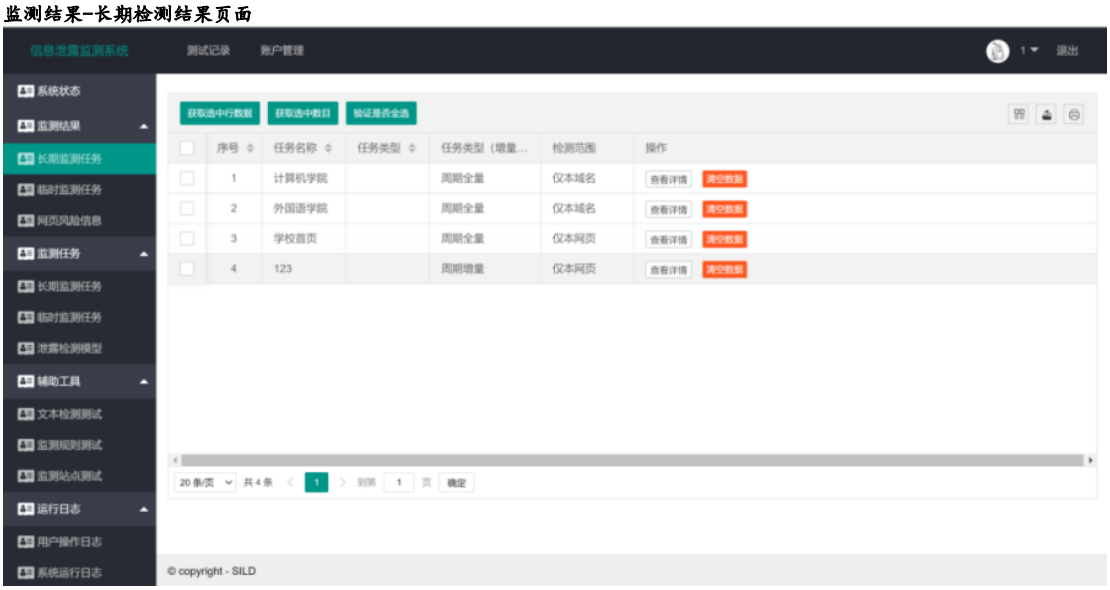
如图5所示，泄露检测页面提供'配置检测模型'和"配置正则规则"两个选项卡，点击配置正则规则，会显示当前已有的正则规则。点击"删除"按钮，则会删除对应的正则规则。点击"增加规则"出现如图6所示页面，输入想要添加到规则名称和正则表达式，点击确认提交便可在图5中显示。点击"规则测试"，即可跳转到文本检测测试页面(图14）。此页面可以帮助查看输入的正则表达式是否能匹配出想要的结果。



点击"配置检测模型"选项卡，如图7所示。显示已有的检测模型，可以在此页面增加系统中的检测模型。点击查看详情，跳转至图8所示界面。此界面显示该检测模型对不同信息组合给出的风险等级。用户可以更改某个风险模型中不同关联规则对应的风险等级，也可以增加改模型中的关联规则项。

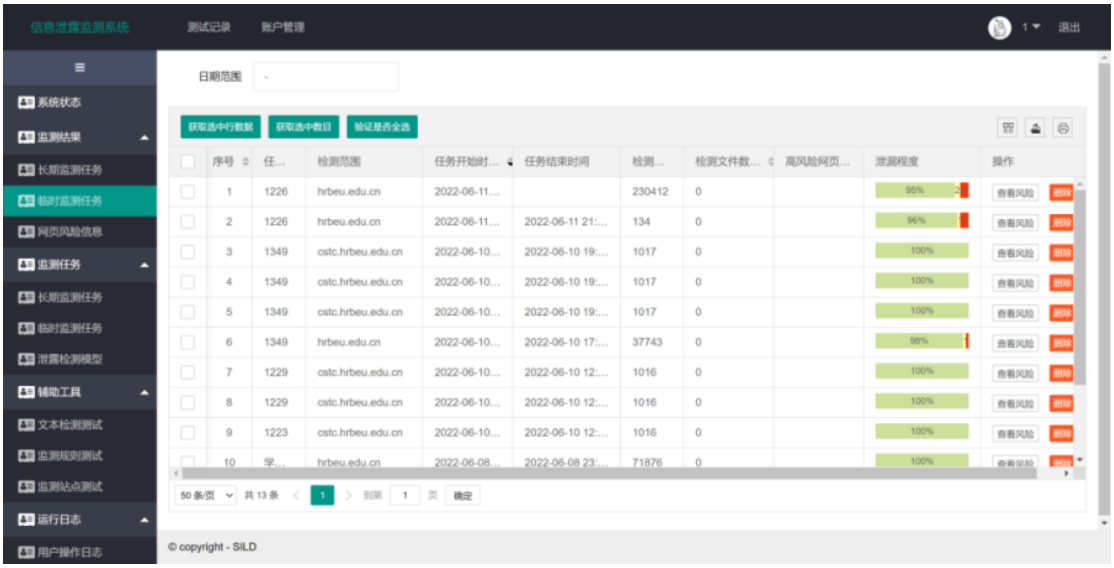


监测结果-长期检测结果页面

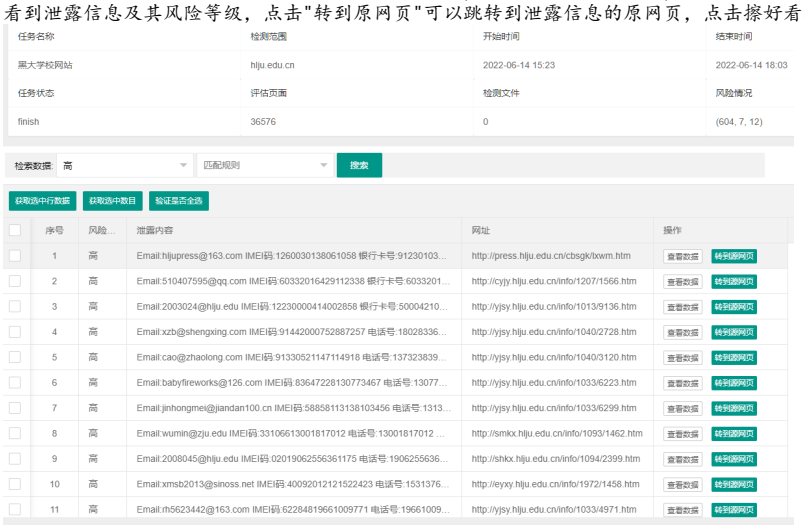


如图9所示，监测结果-长期监测任务页面是监测任务-长期监测任务执行结果对应的页面。页面显示相应的监测任务名称、类型、范围。点击"查看详情"，可以查看此监测任务监测出泄露的信息

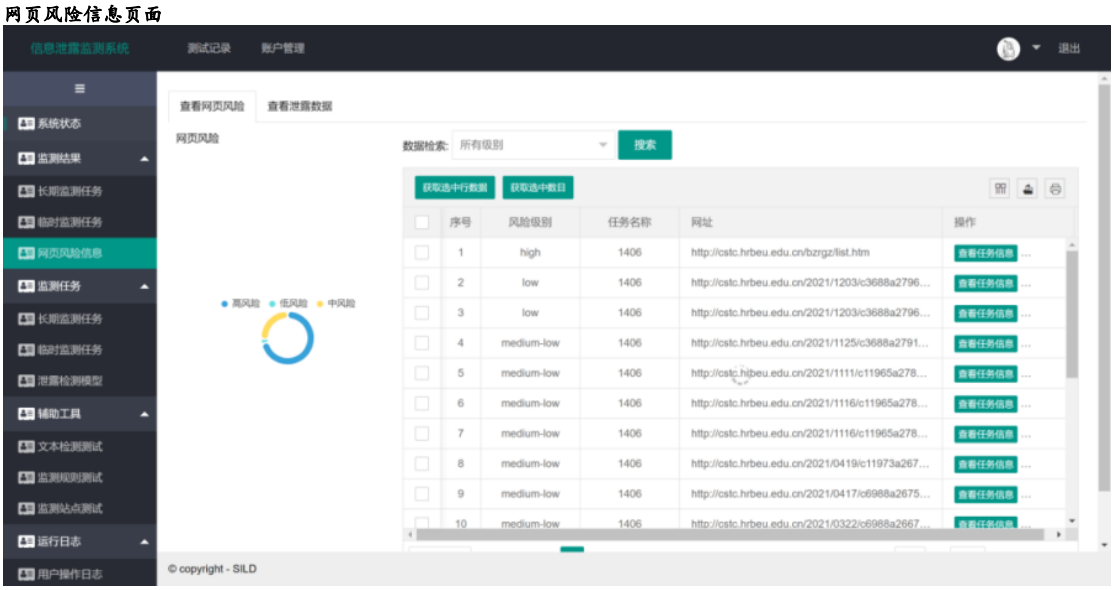
监测结果-临时监测任务界面



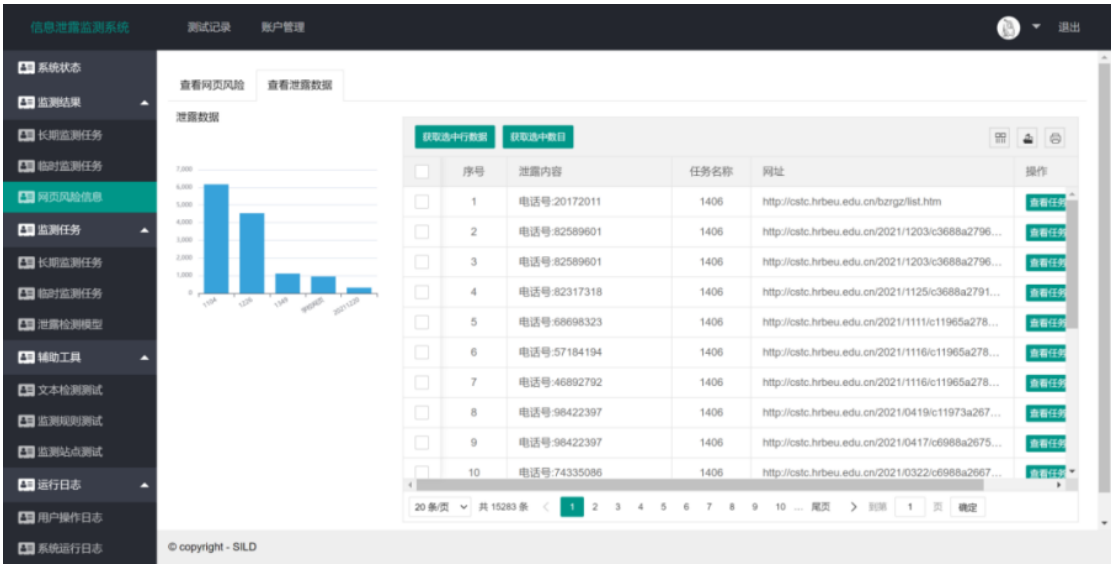
如图10所示，监测结果-临时监测任务页面是监测任务-临时监测任务执行结果对应的页面。页面显示对应任务的名称、检测范围、检测页面数量、文件个数、高风险网页数量、泄露程度等。点击"查看风险"，显示如图11所示界面，该页面显示了此任务泄露的详细信息。可以从"泄露内容"和"风险"对应栏中查看到泄露信息及其风险等级，点击"转到原网页"可以跳转到泄露信息的原网页，点击擦好看数据可以查看系统用于风险评估的原始信息。



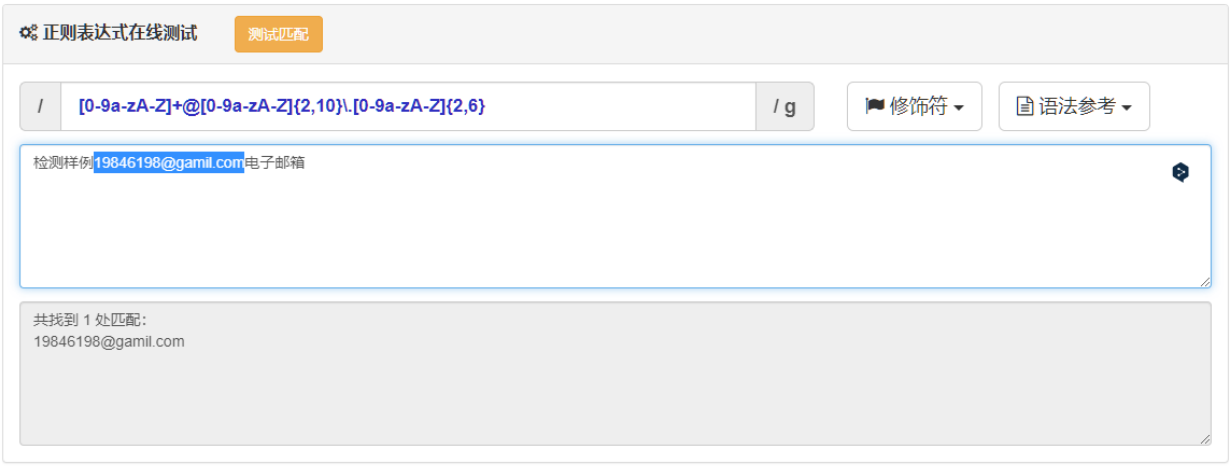
网页风险信息页面



网页风险信息页面如图12所示，点击"查看网页风险"选项卡，显示各个检测任务所检测网页对应的风险级别，并对统计了表格中所有中高低风险数量，将比例情况显示在图形中。点击"查看泄露数据"选项卡，显示如图13所示界面。此界面显示各个任务所检测到网页的泄露内容。并统计表格中任务名称出现的次数，取前5，显示在左侧图形中。



文本检测测试界面



文本检测测试界面如图14所示，此页面有助于查看正则表达式的有效性。在输入正则表达式后，输入待匹配文本，便可查看匹配结果。

检测规则测试界面

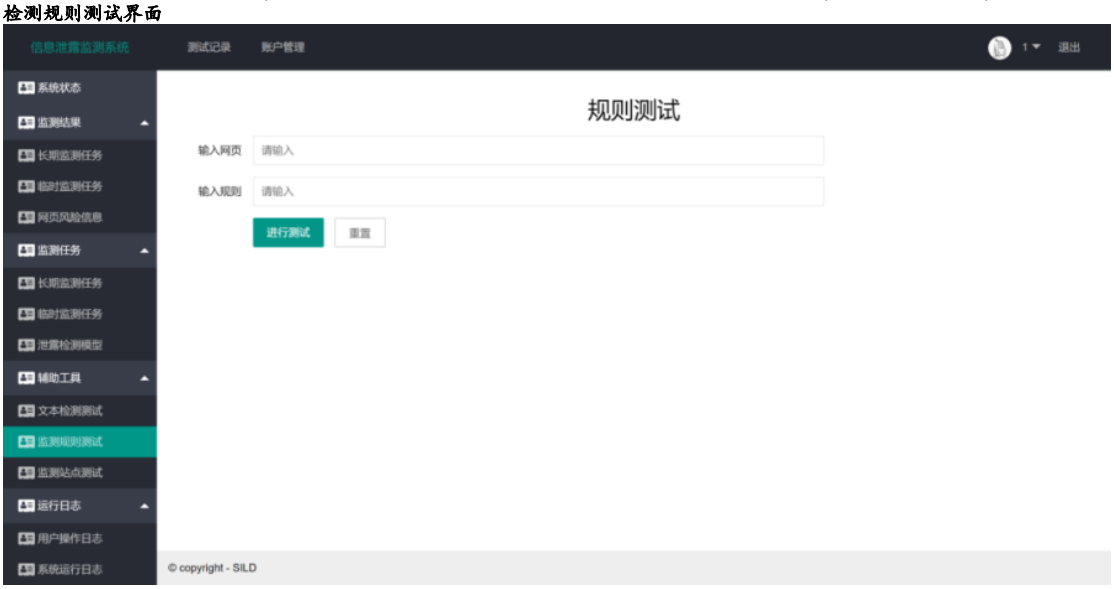
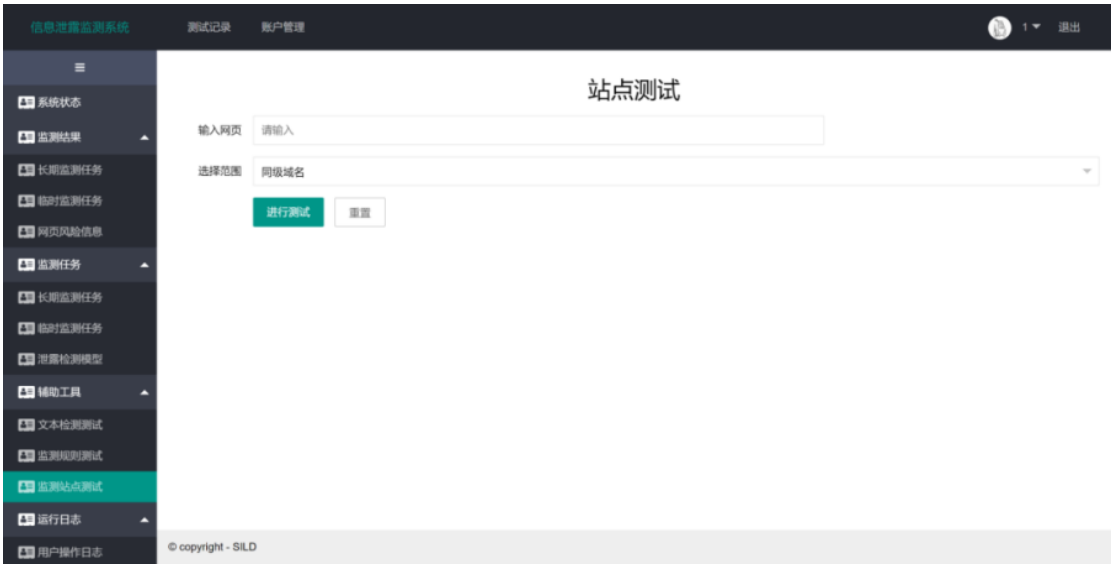


图15

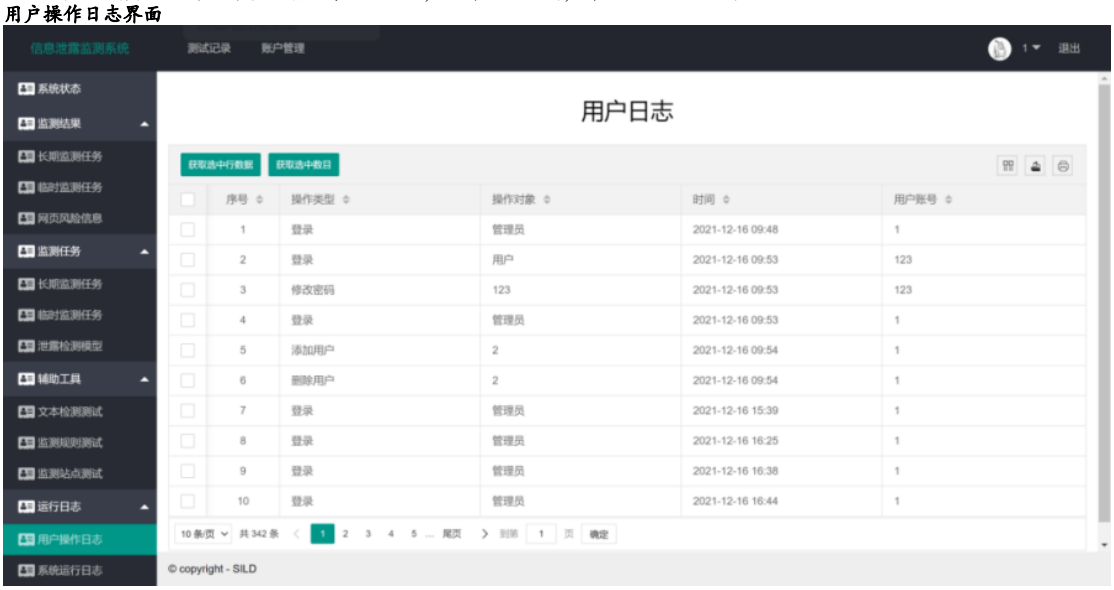
如图15所示，检测规则测试界面。输入待测试网址以及规则。便可开始对该网页进行检测，并将此次测试保存在测试记录中。

监测站点测试



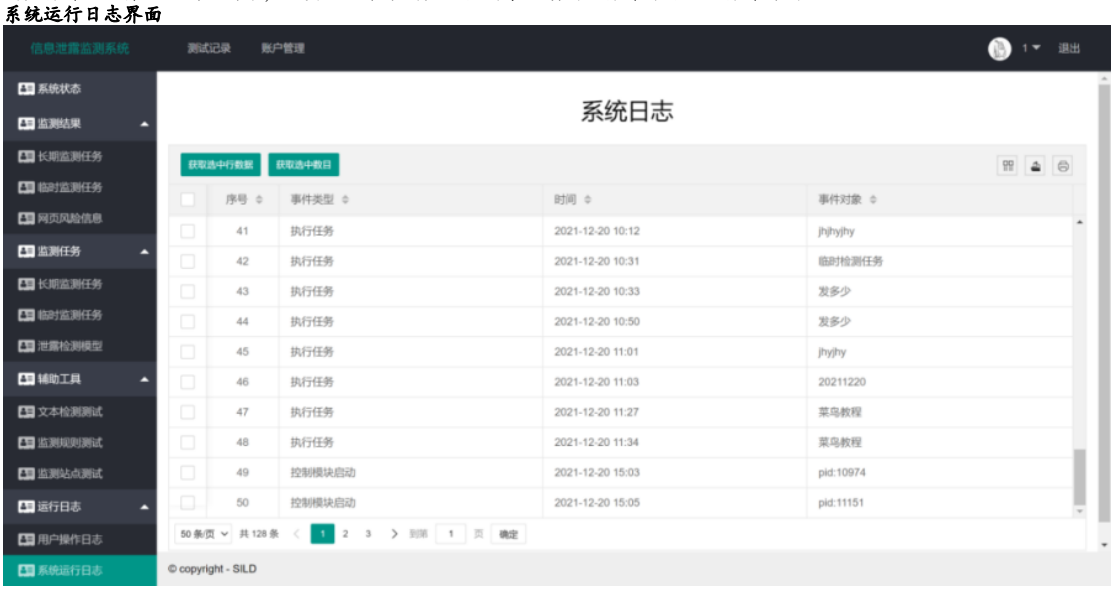
监测站点测试页面如图16所示。输入待检测网址，选择检测范围，即可对此网址进行检测。

用户操作日志界面



用户操作日志界面如图17所示，该页面记录了用户的使用系统期间的操作行为以及操作时间。

系统运行日志界面



系统运行日志界面如图18所示，系统日志记录了系统运行期间自身运行情况。

五、创新与特色

说明作品在创意、技术、应用或设计等方面的创新与特色点，限三至五项。

1. 在当今疫情时代下，个人信息大量出现在网页和文件中，而系统的功能是评估网页及其附件中的个人隐私信息泄露风险，通过技术手段预先筛选出具有个人信息泄露风险的网页，并且可以通过设置周期监测，对网页进行持续监控，有利于组织机构对其网页进行评估，及时发现网页中的隐含的个人信息泄露风险，是贴合时代背景的应用型创新。
2. 基于《中华人民共和国数据安全法》和《中华人民共和国个人信息保护法》以及相关法律法规中对个人隐私信息的定义，进行泄漏风险评估模型的建立，各数据项组合对应泄露风险的制定有法可循，更加能体现出数据安全是数字经济发展的压舱石，是国家安全的重要组成部分，数据安全已成为事关国家安全和经济社会发展的重大议题这一精神，有利于推动数据安全，促进数据开发利用，维护公民、组织的数据安全和合法利益。并且系统中的泄漏风险评估模型可以根据用户需求进行调整，以匹配不同用户对不同来源数据存在的差异化评估风险的需求，使得系统的适应性得到加强。
3. 系统根据实际的网络需求，可以分布式部署数据获取程序，多个程序共同对同一范围的网页和页面附件信息进行下载，可以提高数据的获取速度，同时有助于解决数据孤岛情况下，部分网页的访问限制带来的不可检测问题。并且数据获取程序支持断点恢复，在一个数据获取程序中断的同时不影响其他数据获取程序，关闭全部数据获取程序，也可以借助断点恢复重启任务。利用消息队列实现待检测数据的暂存和分发，在一定程度上保证了程序的健壮性，也有利于对数据获取和数据检测进行性能调整，保证了待检测数据不会因为程序问题而丢失，系统支持多个检测程序利用不同模型进行信息检测，数据获取程序和检测程序可以处在不同网段，有助于保证数据在传输和检测过程中的安全性。

附注：以上模板供各参赛团队参考，必须包括但不限于上述内容，标题可适当修改。斜体字部分为说明文字，请自行删除。

视频流程：

- 系统总体功能概述（以导航栏展开）
- **发起任务和查看任务结果**
- **配置检测模型**
- 周期任务、测试功能、辅助功能
- 结语