郑州大学毕业设计（论文）

题 目：基于动态分析的分布式恶意

代码检测系统的设计与实现

指导教师： 庞海波 职称： 讲师

学生姓名 ： 赵晓宁 学号： 20137740467

专 业 ： 计算机科学与技术(软件测试)

院（系）： 软件与应用科技学院

完成时间 ： 2017.04.22

2017年 05月 20

# 摘要

随着电脑手机的普及和互联网的不断发展，我们在生活中和工作中越来越离不开它们。计算机病毒和木马也越来越多的影响我们的生活。面对日益增多的海量恶意代码，传统的检测恶意代码的方式已不能满足检测的需求，因此急需一种能短时间快速检测大量恶意代码并获取结果的方式。本文的主要工作体现在两个方面：

（1）提出使用动态分析技术对恶意代码进行检测。分析系统会将恶意代码和分析模块动态上传至虚拟机然后运行样本文件注入模块。

（2）提出使用分布式集群对恶意代码进行检测。集群中的分支节点可以提供各种类型的客户机，当有大量恶意软件到来时调度系统可以创建队列，为每个恶意代码检测任务分配分析节点和对应类型的客户机，动态开启客户机运行恶意软件，记录其行为。

**关键词**：分布式 负载均衡 动态分析

# Abstract

With the popularity of computer phones and the continuous development of the Internet, we in life and work more and more inseparable from them. Computer viruses and Trojans are also increasingly affecting our lives. In the face of increasing mass malicious code, the traditional way of detecting malicious code has been unable to meet the testing needs, so there is a need for a short time to quickly detect a large number of malicious code and get the results of the way. The main work of this paper is embodied in two aspects:

(1) The use of dynamic analysis technology to detect malicious code. The analysis system dynamically uploads the malicious code and analysis module to the virtual machine and then runs the sample file injection module.

(2) Propose the use of distributed clusters to detect malicious code. The branch nodes in the cluster can provide various types of clients. When a large number of malware arrives, the scheduling system can create queues, assign the analysis nodes for each malicious code detection task, and dynamically switch the client to run the client Software that records its behavior.

**Key words:** distributed load balancing dynamic analysis

# 目录

[摘要 I](#_Toc13808)

[Abstract I](#_Toc7045)

[第1章 引言 1](#_Toc20307)

[1.1 系统背景 1](#_Toc30349)

[1.2 系统开发目标 1](#_Toc24274)

[1.3 论文结构安排 1](#_Toc12889)

[第2章 系统设计技术及开发工具 2](#_Toc24629)

[2.1 系统设计技术概述 2](#_Toc13561)

[2.1.1 分布式 2](#_Toc21648)

[2.1.2集群 2](#_Toc12625)

[2.1.3动态分析 2](#_Toc26864)

[2.2 项目设计工具 2](#_Toc5923)

[2.2.1 Python 2](#_Toc4426)

[2.2.2 Flask 2](#_Toc18630)

[2.2.3 Django 2](#_Toc21483)

[2.2.4 Mongodb 2](#_Toc29536)

[2.2.5 PyCharm 3](#_Toc4423)

[2.3 本章小结 3](#_Toc17901)

[第3章 分布式恶意代码检测系统需求分析 4](#_Toc13145)

[3.1 系统功能需求 4](#_Toc26024)

[3.2 系统数据需求分析 5](#_Toc18678)

[3.2.1 主节点的数据需求 5](#_Toc12939)

[3.2.2 任务的数据需求 5](#_Toc29886)

[3.2.3 分析结果的数据需求 6](#_Toc17666)

[3.3 本章小结 6](#_Toc5264)

[第4章 分布式调度系统模块设计 7](#_Toc5640)

[4.1 分布式调度系统设计原则 7](#_Toc28078)

[4.2 项目总体框架体系 7](#_Toc3389)

[4.3调度模块设计 7](#_Toc19192)

[4.4 主节点restful api设计 8](#_Toc19652)

[4.5 从节点restful api设计 9](#_Toc9688)

[4.6分布式模块Web界面设计 9](#_Toc4321)

[4.7 数据库总体设计 9](#_Toc30694)

[4.8 本章小结 10](#_Toc27054)

[第5章 分布式调度系统实现 11](#_Toc580)

[5.1主节点restful api实现 11](#_Toc1893)

[5.2从节点restful api实现 12](#_Toc324)

[5.3web界面实现 14](#_Toc22034)

[5.4数据库实现 16](#_Toc13785)

[5.5 本章小结 17](#_Toc19362)

[第6章 分布式恶意代码检测系统的测试 18](#_Toc15320)

[6.1系统测试原则 18](#_Toc9138)

[6.2系统功能测试 18](#_Toc6946)

[6.3本章小结 20](#_Toc24200)

[结论 21](#_Toc13092)

[参考文献 22](#_Toc17689)

[致谢 23](#_Toc23378)

# 

# 1 引言

## 1.1 系统背景

现今，互联网和计算机发展越来越快，我们在工作和生活中越来越依赖电脑，我们使用它们购物，支付，玩游戏，聊天。但是网络安全问题始终如影随形。由于接触互联网的人越来越多，病毒和木马的制作已经越来越容易被人们掌握。因此恶意代码的数量呈几何增长，之前靠单机就可以处理恶意代码的方法已经不能应对海量的分析需求。因此，急需一种能快速检测海量恶意代码的系统来自满足这种需求。

## 1.2 系统开发目标

本系统最终达到的目标是当有队列中有大量的恶意代码检测任务时，主节点调度模块可以将各个分析任务负载均衡分配到每个空闲的分析节点，并在该节点启动分析任务并获取分析结果，最终将各个节点上的分析结果汇总至主节点进行统一处理。

## 1.3 论文结构安排

论文共由6个章节组成，主要内容及结构安排如下：

第1章绪论，介绍本课题的背景来源、研究意义和主要研究内容，以及该系统开发的最终目标。

第2章系统设计技术及开发工具，介绍了本系统用到的各种技术，以及开发过程中用到的各种工具和框架。

第3章分布式恶意代码检测系统需求分析，介绍了本系统本系统在设计过程中的总体需求分析，以及功能需求分析和数据需求分析。

第4章分布式调度系统模块设计，主要介绍了分布式调度系统的设计原则，项目总体框架体系设计，以及各个子模块的设计，包括调度模块设计，主节点restful api，从节点restful api，web界面和数据库的总体设计。

第5章分布式调度系统实现，主要介绍了分布式调度系统是如何用代码实现的。包括主节点restful api的实现，从节点restful api的实现，web界面的实现以及数据库的实现。

第6章分布式恶意代码检测系统的测试 ,记录了对本系统进行测试的过程，首先介绍了系统测试的原则，然后是对系统的各个模块进行功能测试，包括主节点restful api的测试，从节点restful api的测试，以及web界面的测试。

# 2 系统设计技术及开发工具

## 2.1 系统设计技术概述

2.1.1 分布式

中心化的主要问题是可靠性。如果中央节点关闭，则整个系统不可用[1]。除了解决一些中心化存在的问题外，分布式往往会将任务分配到每个子系统，但是分配会带来很多其他的问题[2]。主要是一致性。简单来说，分布式系统中每个子系统承担相同任务的不同部分，在调度进程控制下，一起运行以完成相同的任务[3]。分布式处理系统包括硬件，控制系统，接口系统，数据，应用等要素。

2.1.2集群

集群技术是一种今年来新兴起的技术，用户通过集群技术，可以将分散的多台机器综合起来进行统一管理，以发挥这些机器的最大性能。使用集群技术可以让用户在付出较低代价的情况下获得相对较高的收益。

2.1.3动态分析

动态分析是对恶意代码的行为进行分析，静态分析是将可执行文件以二进制形式打开[4]，根据pe结构中所定义的规范找到可执行文件的代码块，数据块，导入表，导出表等信息[5]，将这些信息静态的从二进制文件中取出展示给分析者，让分析者可以看到它们有哪些可疑的操作。有些可执行文件加密之后，如果只使用静态分析是不能看到它们的可疑操作的[6]，只有在运行时它们才会进行那些有害的操作。而动态分析则是将可执行文件在真实的系统或者沙箱中运行，记录其进程之间的调用关系以及调用了哪些函数，该可执行文件对磁盘的操作以及该可执行文件运行时的流量等信息。

## 2.2 项目设计工具

2.2.1 Python

Python并不是一种新出现的计算机编程语言，它具有面向对象和解释执行等特点，它是于1989年由荷兰语Guido van Rossum发明[7]，于1991年公开发布第一个发行版。它的语法简洁扼要，Python有丰富而强大的库。

2.2.2 Flask

flask是一个使用Python编写的轻量级Web应用框架。用户在使用时只需要导入相关的包，在代码中为链接和函数建立映射关系，之后启动该python代码文件便可以轻松创建web服务器[8]。

2.2.3 Django

Django是一个开源cms的Web应用框架，由Python编写。采用了MVC的框架模式。默认为使用者提供了一个后台管理界面，只需要用户设计好数据库，编写好相对应的实体类便可自动为用户生成对这些数据表的增删改查方法的页面[9]，而用户只需要完成前台页面的编写。

2.2.4 Mongodb

MongoDB是基于分布式存储的非关系型数据库。它采用C ++语言编写而成。目的是为开发人员提供一种可扩展的，高性能的数据存储解决方案。它是一种介于关系型和非关系型数据库之间的一种产品，它的每条记录的格式类似于json，所以他可以存储更加复杂的数据类型[10]。而且它的查询语言也非常强大，其语法有些类似于面向对象的查询方式，而且还支持数据索引[11]。

2.2.5 PyCharm

PyCharm是一个Python IDE，它提供一系列工具，可以让用户高效的编写python代码。它提供代码调试，对项目进行管理，版本控制，智能提示，代码跳转，单元测试等功能，并且支持django，flask等框架的python应用的开发。现在pycharm已经成为python专业开发人员不可或缺的强大工具。

## 2.3 本章小结

本章主要介绍了系统设计过程中的一些技术概念和项目设计用到的一些工具，技术包括分布式，集群，动态分析。项目设计用到的工具包括Python，flask，django，mongodb，postfresql和pycharm。

# 3 分布式恶意代码检测系统需求分析

## 3.1 系统功能需求

该系统需要让用户可以通过restful api或者web界面提交恶意代码样本至主节点的数据库中，调度系统将会为每个任务分配其请求的资源（虚拟机的类型，版本以及网络条件等），当样本的数量足够多时可以将它们负载均衡到集群里面每个节点上进行分析，并将结果返回到主节点。需要为用户提供的功能包括提交任务，注册节点，删除节点，获取任务状态，获取任务列表，查看任务详细信息获取节点信息等功能。

系统部署在云平台上如图3.1所示，包括一个主节点两个从节点，主节点负责调度任务，两个从节点中每个从节点的virtualbox中都有两台客户机。用户从主节点的web界面提交四个任务后，这些任务被负载均衡到每个从节点（每个从节点的每个客户机都被分配到一个任务）。从节点的调度进程会启动该节点的每个客户机并且开始分析任务。

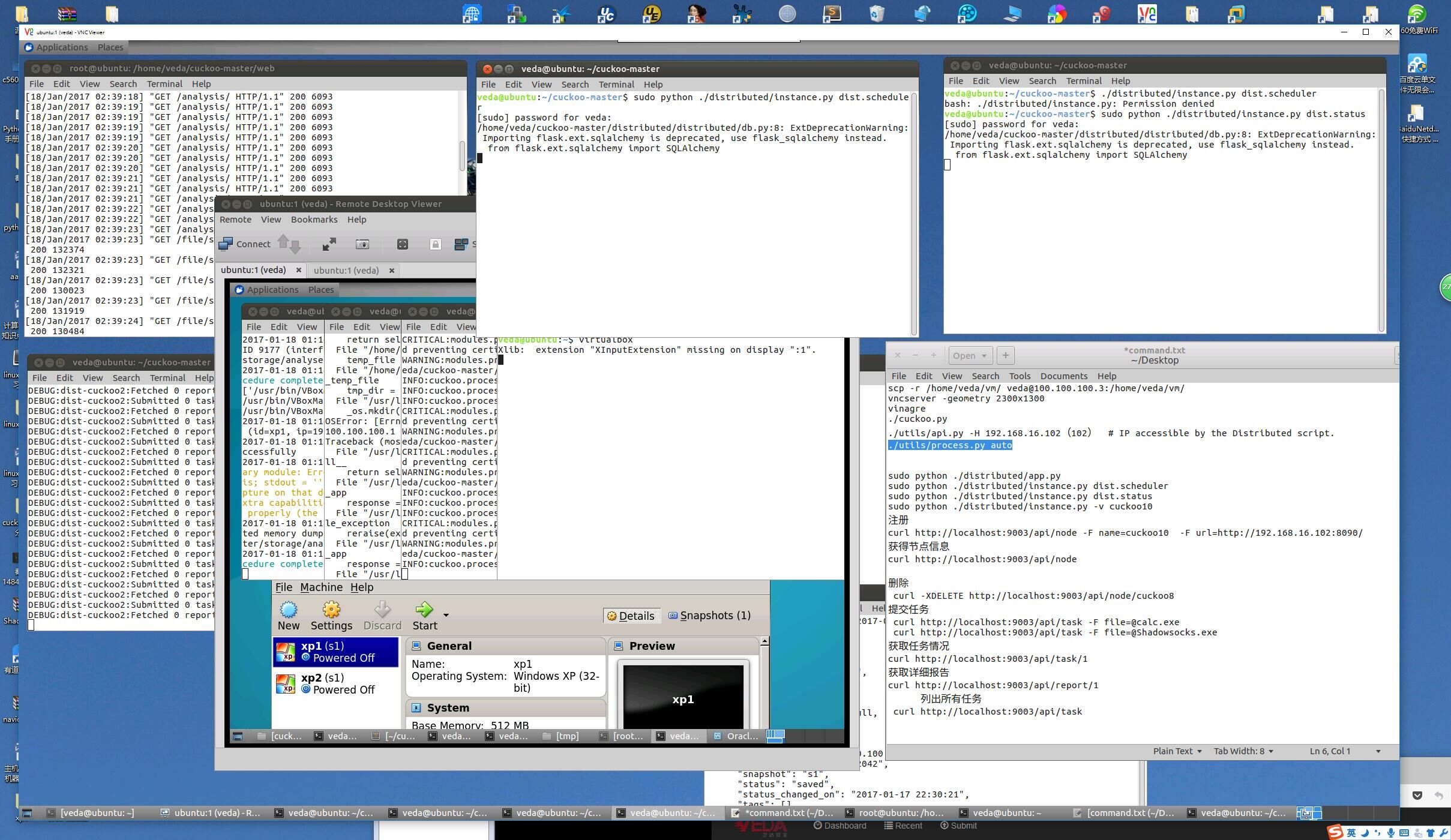


图3.1 系统部署在云平台上的截图

功能需求用例图如图3.2所示：

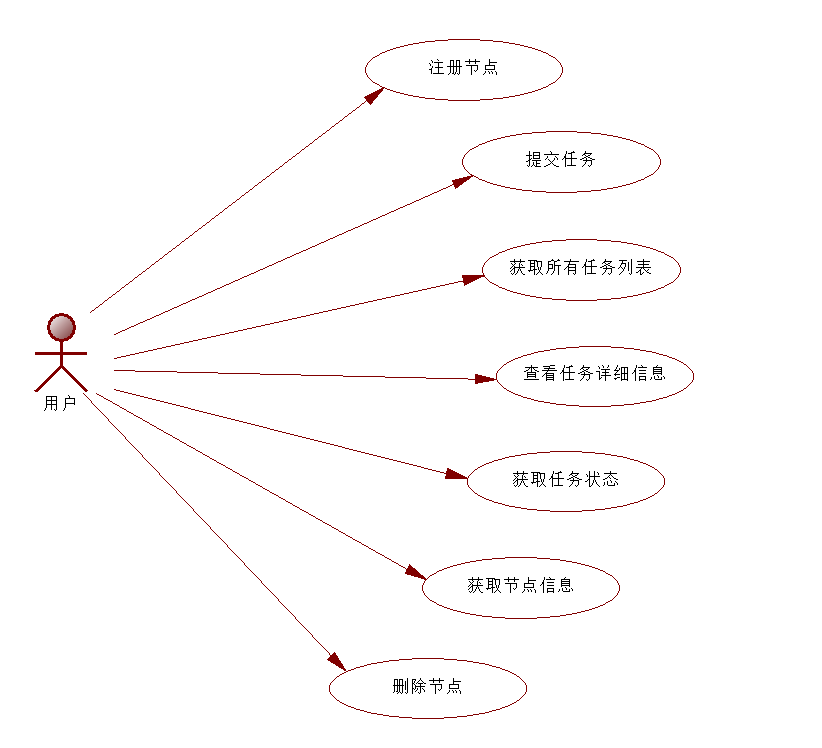


图3.2 系统总体需求用例图

## 3.2 系统数据需求分析

该系统对数据的需求主要体现在三个方面：主节点的数据需求，任务的数据需求和分析结果的数据需求。

3.2.1 主节点的数据需求

主节点的数据需求体现在两方面：注册的从节点的各种属性，包括节点的id，节点名字，节点的url，节点的激活状态，节点的模式。客户机的各种属性，包括客户机的id，客户机的名字，客户机的平台类型（windos，linux，其他等），客户机的标签，客户机的所属节点id。他们都需要通过Alchemy 框架存储在postgresql数据库中。

3.2.2 任务的数据需求

任务的数据需求体现在任务的各种属性，包括任务的类型，请求的虚拟机的类型，超时时间，任务的状态（包括等待中，正在进行中，错误，已完成，已报告等），任务的优先级，任务的所有者，任务分配的从节点的id，任务的提交时间，任务的删除时间，任务的完成时间，任务对应的样本的md5，任务的样本文件的存储路径，任务的样本文件的类型等，他们都需要通过Alchemy 框架存储在mysql数据库或postgresql数据库中。

3.2.3 分析结果的数据需求

分析结果的数据需求体现在分析结果的各种属性，包括分析样本的静态分析结果（包括样本文件的数据块数据，样本文件的代码块数据，样本文件的导入表和导出表），分析样本运行时的截图，分析任务运行时网卡的数据（包括各种协议的数据包），分析样本的动态分析结果（包括进程的调用关系，函数的调用信息，分析样本对磁盘的操作等），因为每个分析任务所对应的记录的容量巨大并且每个分析任务需要的分析模块是不一样的所以分析结果的大小是不确定的因此需要将结果存储在mongodb中。

## 3.3 本章小结

本章主要对分布式恶意代码监测系统的需求分析进行了描述，包括系统功能需求分析，系统数据需求分析两大部分。

# 4 分布式调度系统模块设计

## 4.1 分布式调度系统设计原则

系统设计是根据系统分析阶段确定的系统的逻辑模型和功能要求，应该遵循开放性原则，能够支持多种网络系统，支持二次开发。系统应采用标准数据接口，具有与其他信息系统进行数据交换和数据共享的能力。设计的系统要操作简单，实用性高，具有易操作、易维护的特点。并且，系统应具备在出现故障时，能得到及时、快速地进行自维护。在满足系统功能及性能要求的前提下，尽量降低系统建设成本，采用经济实用的技术，综合考虑系统的建设、升级和维护费用。系统应符合向上兼容性、向下兼容性、配套兼容和前后版本转换等功能。系统采用的协议应符合国家标准、行业标准和技术规范。系统应具有良好的兼容性和互联互通性。

## 4.2 项目总体框架体系

分布式调度模块主要分为主节点和从节点两大部分，主节点负责接收任务，调度任务和汇总任务的结果。从节点负责接受主节点的调度和运行主节点为他分配的任务，并将分析的结果传回主节点。

系统的运行也分为两部分。首先是从节点的启动，在系统开机后首先需要运行virtualbox虚拟机创建vboxnet0虚拟网卡。然后运行cuckoo.py文件，该文件启动之后会创建服务器，加载各个分析模块，获取该节点当前可用的客户机，并周期性的查询数据库的tasks表，取出所有的任务并筛选出status为pending的任务[12]，如果有处于等待状态的任务则取出该任务，并准备好该任务所需要的条件（将样本文件，分析所需要的python脚本和dll动态链接库打包成一个zip文件）然后开启虚拟机，恢复该虚拟机之前保存的快照（该快照运行了一个agent.py脚本创建了一个服务器[13]，调度程序会去连接该服务器并将打包好的文件上传至虚拟机，然后该脚本将收到的zip压缩文件解压到之前配置的目录。），虚拟机启动之后该进程会发送指令运行各个模块（包括模拟鼠标点击，截图等），并让虚拟机启动样本文件并注入之前准备好的动态链接库（该动态链接库会hook一些系统敏感的函数[14],当测试样本调用这些函数时它会记录这些函数的调用情况,并连接agent.py创建的命名管道将收集的的信息发送至agent进程）[15]，agent进程会连接cuckoo.py文件创建的服务器并将收集到的结果实时传递给它，最后由它将结果存储至mongodb和磁盘。最后还需要启动utils/api.py它会创建一个服务器来供主节点连接为主节点服务。

然后是主节点的启动，在主节点的系统启动需要启动四个服务。第一个是web服务。Web服务采用python django框架，用户需运行web/mange.py文件来启动服务。第二个是供用户在控制台使用restful api提交任务的服务它采用flask框架实现，对用户提供了节点注册，节点删除，获取节点的信息，提交任务，获取任务状态，列出所有任务等功能。需要运行distribute/app.py来启动。第三个是调度服务，用户需运行distribute/instance.py文件来启动调度服务，该服务包括三部分：分配任务，设置节点状态，监控每个可用的节点。

## 4.3调度模块设计

调度模块主要有三个部分：

（1）分配任务：

找出所有可用节点，查询节点的状态，找出处于等待状态的任务，将任务分配到查询出来的空闲节点（设置任务的节点id）。

（2）设置节点状态：

每隔一段时间向所有存活节点发送心跳根据心跳回应获取该节点此时的状态，然后将节点的状态保存至数据库。

（3）监控每个可用的节点：

从数据库取出该节点所有分配到的处于等待状态的任务，将任务使用restful api远程提交至该节点。并周期性从该节点查询出任务列表，如果有处于完成状态的任务则通过restful api从该节点获取该任务的报告（分析结果包括截图等信息的压缩包），完成之后再删除该任务。

调度模块时序图如图4.1所示：

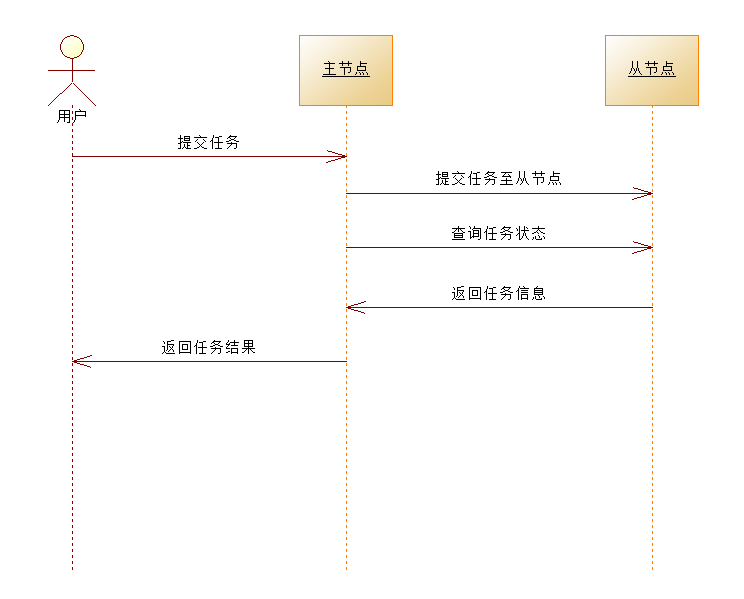


图4.1 调度模块时序图

## 4.4 主节点restful api设计

主节点restful api使用python flask框架设计，主要包括：

任务提交

节点注册

节点删除

获取节点的信息

提交任务

获取任务状态

列出所有任务

## 4.5 从节点restful api设计

从节点restful api使用python flask框架设计，主要包括：

创建任务

提供此节点任务列表

提供此节点某个任务的状态

提供任务报告

提供任务的pcap包

提供此节点状态

提供此节点的客户机信息

## 4.6分布式模块Web界面设计

Web界面采用python django框架，用户可通过web界面提交任务和查看结果。web界面包括三个标签：任务统计，查看历史任务和提交任务；

任务统计界面：

显示历史任务的总个数，样本的个数以及所有任务的状态统计。

历史任务界面：

显示分析过的所有任务的列表，点击对应的id可以查看该任务的详细信息（包括恶意代码运行时的截图，进程的调用关系，代码执行时的流量信息，静态分析结果，恶意代码运行时的行为）。

提交任务界面：

用户可以选择提交任务的种类文件还是url，或者apk，选择任务的超时时间以及网络。

## 4.7 数据库总体设计

分布式模块数据库采用postgresql和mongodb，因为每次分析所使用的模块可能不同所以分析结果是不固定的因此使用mongodb存储每个任务的分析结果。

其他数据用postgresql数据库存储，主要有三张表task，node和machine。

Task表负责任务的添加分配以及结果的获取，其结构如表4.1所示：

表4.1 task表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名称 | 名称 | 属性类别 | 长度 | 空值 | 键类 |
| ID | id | int | 11 | Null | Yes |
| Path | Username | varchar | 255 | Null |  |
| Filename | Password | varchar | 255 | Null |  |
| Package | Email | varchar | 255 | Null |  |
| Timeout | Sex | varchar | 255 | Null |  |
| Priority | Phone | varchar | 255 | Null |  |
| Options | Icon | varchar | 255 | Null |  |
| Machine | nikename | varchar | 255 | Null |  |
| Platform | last\_login\_time | varchar | 255 | Null |  |
| Tags | login\_count | varchar | 255 | Null |  |
| Custom | registered\_time | varchar | 255 | Null |  |

续表4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Owner | realname | varchar | 255 | Null |  |
| Memory | safe\_q\_1 | varchar | 255 | Null |  |
| Clock | safe\_q\_2 | varchar | 255 | Null |  |
| Enforce\_timeout | safe\_q\_3 | varchar | 255 | Null |  |
| Node\_id | safe\_a\_1 | varchar | 255 | Null |  |
| Task\_id | safe\_a\_2 | varchar | 255 | Null |  |
| status | safe\_a\_3 | varchar | 255 | Null |  |
| submitted | submitted | timestamp |  | Null |  |
| Deleted | Deleted | timestamp |  | Null |  |
| Started | Started | timestamp |  | Null |  |
| completed | completed | timestamp |  | Null |  |
| Md5 | Md5 | varchar |  | Null |  |
| category | category | varchar |  | Null |  |

Node表负责节点的创建删除，其结构如表4.2所示：

表4.2 node表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名称 | 名称 | 属性类别 | 长度 | 空值 | 键类 |
| ID | id | int | 11 | Null | Yes |
| Name | name | varchar | 255 | Null |  |
| url | url | varchar | 255 | Null |  |
| Mode | mode | varchar | 255 | Null |  |
| Enabled | enabled | varchar | 10 | Null |  |

Machine表负责展示每个节点拥有的客户机其结构如表4.3所示：

表4.3 machine表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名称 | 名称 | 属性类别 | 长度 | 空值 | 键类 |
| ID | id | int | 11 | Null | Yes |
| Name | name | varchar | 255 | Null |  |
| Platform | Platform | varchar | 255 | Null |  |
| Tags | Tags | varchar | 255 | Null |  |
| Node\_id | Node\_id | Integer | 10 | Null |  |

## 4.8 本章小结

本章介绍了分布式调度系统的设计原则，项目的总体框架体系以及分布式调度系统的各个模块是如何设计的和它们用到的技术，包括主节点restful api的设计，从节点restful api的设计，web界面的设计以及数据库的设计。

# 5 分布式调度系统实现

## 5.1主节点restful api实现

主节点的restful api通过采用Python flask框架来实现，用flask通过修饰器对路径distribute/distribute/views/api.py将文件和视图函数进行了映射主要有：

任务提交：

通过@app.route("/task", methods=["POST"])建立映射，用户向http://url/api/task发送post请求，并提供任务提交时的参数，包括提交的样本文件，以及任务所需要的package，timeout，priority，options，machine，platform，tags，custom，owner，memory，clock，enforce\_timeout等信息，服务器收到该请求后调用建立任务方法，将传递过来的各种参数保存至数据库的task表中，并返回创建任务成功的json字符串。

节点注册：

通过@blueprint.route("/node", methods=["POST"])建立映射，用户向http://url/api/node发送post请求，并且提供节点的name和url。服务器收到请求后先查询数据库的node表，判断节点的名称是否之前已经被注册过，如果未被注册过则向从节点发送POST请求，获取该节点拥有的客户机的种类，数量以及其他信息如图5.2，并将返回的集合中的每条信息与该node对应起来，最后将这些信息保存在node表和machine表中并返回创建节点成功的json字符串如图5.1所示。

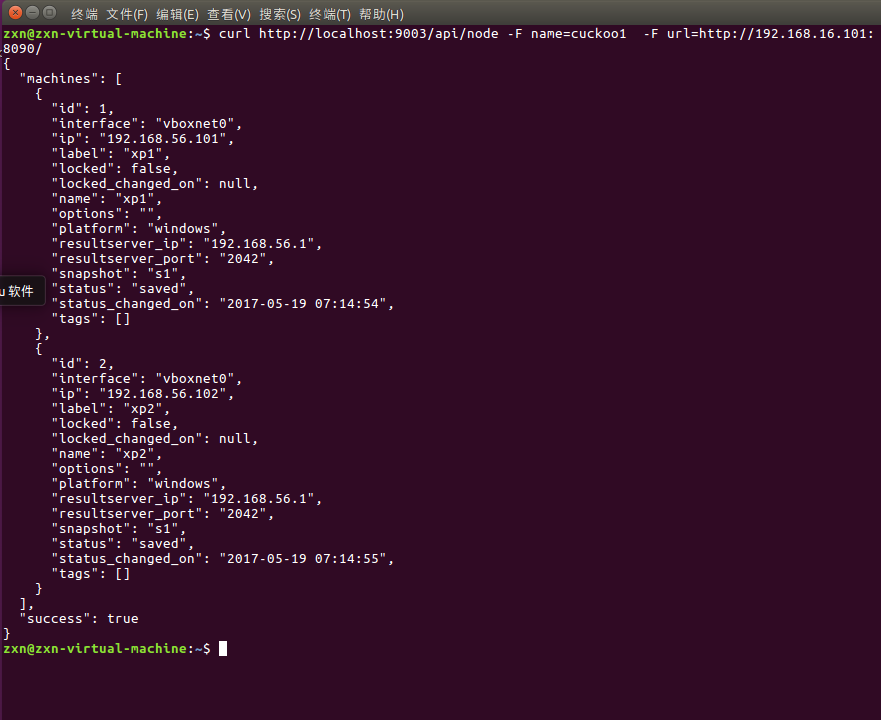


图5.1 注册节点截图



图5.2 从节点返回客户机截图

节点删除：

通过@app.route("/node/<string:name>",methods=["DELETE"])建立映射,用户使用curl向http://url/api/node发送 -XDELETE 请求。服务器收到请求后先判断该节点名称是否在node表中存在，如果不存在则返回404及节点不存在错误，如果存在，并不会删除该节点，而是该节点的enabled状态设置为false最后返回删除节点成功的json字符串。

获取节点的信息：

包括两个映射:通过@app.route("/node")建立映射，获取所有节点的信息，通过@app.route("/node/<string:name>")建立映射,获取指定名称节点的信息。用户向http://url/api/node或http://url/api/node/<string:name>发送get请求。服务器收到请求后先判断name是否为none并从 node表中查询出所有节点的信息，如果name不存在返回节点不存在，如果存在则继续从machine表中查询该节点所拥有的客户机的集合并将得到的信息保存在字典中，最后返回所有节点和其所拥有客户机的json字符串。

获取某个任务状态：

通过@app.route("/task/<int:task\_id>")建立映，用户向http://url/api/task/<int:task\_id>发送get请求。服务器收到请求之后先判断taskid是否存在于task表中，如果不存在则返回404和任务不存在，如果存在则从数据库查询出该id对应的任务的记录，并封装为json字符串返回。

获取所有任务状态：

通过@app.route("/task")建立映射，用户向http://url/api/task发送get请求并传递offset，finished，status，owner等信息。服务器收到请求之后先判断finished和status是否有值，如果不存在则返回404和不存在，如果存在则从数据库查询并筛选出对应的任务的记录，并封装为json字符串返回。

## 5.2从节点restful api实现

主节点的restful api采用Python flask框架来实现，在utils/api.py文件用flask通过修饰器对路径和视图函数进行了映射主要有：

（1）创建任务

通过@app.route("/tasks/create/file", methods=["POST"])建立映射，主节点向http://url/tasks/create/file发送post请求并传递file，package，timeout，priority, options, machine, platform, tags, custom, owner, tid等信息。服务器收到请求之后先将传过来的恶意代码文件存储到磁盘并返回存储的路径，然后将返回的路径以及其他信息存储至mysql数据库并返回创建任务的taskid。

（2）提供此节点任务列表

通过@app.route("/tasks/list")建立映射，主节点向http://url/tasks/create/file发送get请求并传递owner，status等信息，从节点在收到请求后会从数据库中查询出符合条件的任务的列表，并将列表转化为集合，最后将集合转化为json字符串并返回给主节点。

（3）提供此节点某个任务的状态

通过@app.route("/tasks/view/<int:task\_id>")建立映射，主节点向http://url/tasks/view/<int:task\_id>发送get请求，从节点在收到请求后会从数据库查询出该id所对应的任务，如果id不存在则返回404错误和该任务不存在json字符串，如果该id存在则从数据库查询出该id对应的任务，并将结果封装为json字符串返回。

（4）提供任务报告

通过@app.route("/tasks/report/<int:task\_id>")建立映射，主节点向http://url/tasks/report/<int:task\_id>发送get请求并传递需要返回的报告类型，压缩格式等参数。从节点在收到主节点的请求后首先会获取主节点传递过来的各个参数，然后会从数据库查询出该id的对应的任务是否存在，如果不存在则会返回404和该任务不存在json字符串。如果存在则从数据库查询出该任务的结果在磁盘存储的位置，并根据传递来的信息将需要打包的文件（比如该任务运行时的截图，该任务运行时虚拟机网卡的的pcap包，该任务的html报告页面，该任务的json报告结果）打包一个zip文件传递给主节点。

（5）提供任务的pcap包

通过@app.route("/pcap/get/<int:task\_id>")建立映射，主节点向http://url/pcap/get/<int:task\_id>发送get请求。从节点在收到请求后会去磁盘的storage/analysis/id/dump.pcap路径寻找改文件，如果该文件不存在，则返回404和文件不存在json字符串，如果该文件存在则读取该文件，并将response的返回头设置为文件流，最后将读取到的文件返回至主节点。

（6）提供此节点状态

通过@app.route("/cuckoo/status")建立映射，主节点向http://url/cuckoo/status发送get请求，从节点在收到请求后会从数据库取出该节点所拥有的客户机的数量和每台客户机的信息并将信息封装为一个集合，最后将该集合转化为json字符串返回给主节点。

（7）提供此节点的客户机信息

通过@app.route("/machines/list")建立映射，主节点向http://url/machines/list发送get请求，从节点在收到请求后会收集该节点的内存大小，磁盘容量，软件版本，主机名称，等信息，并从数据库查询出该节点的处于等待，正在运行，已经完成和获取报告的状态的任务的数量，最后将获取到的信息封装为json字符串返回给主节点。

## 5.3web界面实现

主节点拥有一个web界面，它采用python django框架实现。首先在url.py文件配置好路径和映射关系，然后在views.py文件对之前配置的映射的路径进行处理，获取用户需要的信息并进行封装最后展示在页面上。它的主要功能有：

（1）任务统计

任务统计界面也是web界面的主界面它在dashboard/urls.py文件中使用url(r"^$", views.index)建立映射，当用户访问主界面时会调用dashboard/views.py文件中的index（）方法，该方法会从数据库筛选出状态为已完成，正在进行中，已删除，等待中，分配中等状态的任务的数量并将它们封装好然后跳转至dashboard/index.html页面如图5.3所示。

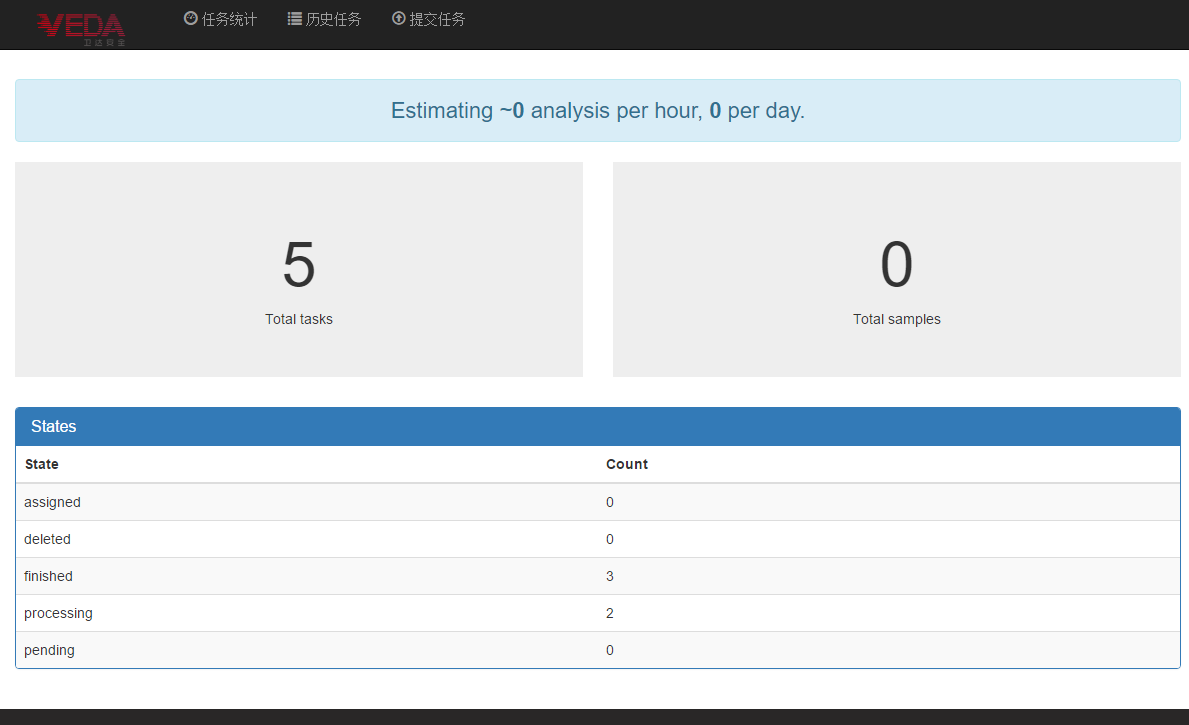


图5.3 任务统计界面截图

（2）历史任务界面

历史任务界面在analysis/urls.py文件中使用url(r"^$", views.index)建立映射，用户点击主界面的历史任务标签时会调用analysis/views.py文件中的index（）方法，该方法会从数据库中task表中根据id大小查询出前50条分析任务，并且根据url和file分好类然后跳转至analysis/index.html页面如图5.4所示。

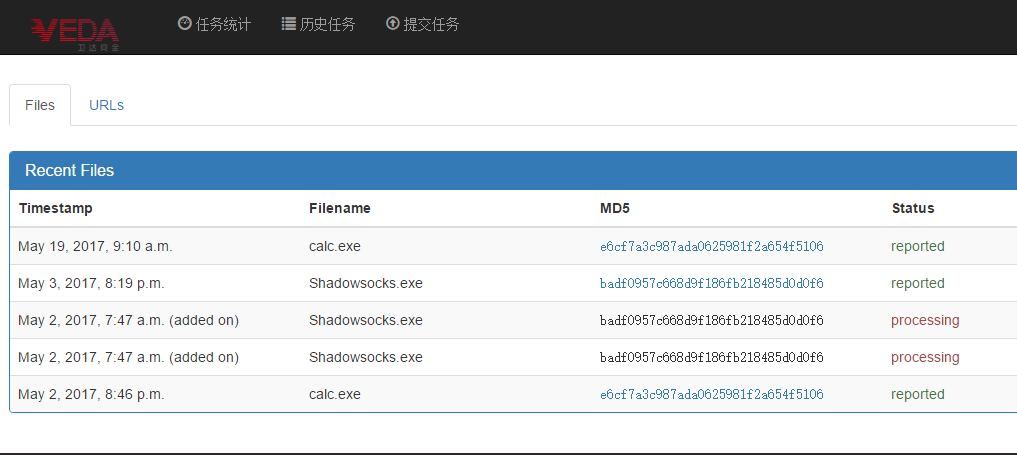


图5.4历史任务界面截图

（3）提交任务界面

提交任务界面在submission/urls.py文件中使用url(r"^$", views.index)建立映射当用户点击主界面的提交任务标签时会调用submission/views.py文件中的index（）方法它首先会判断用户提交的是get还是post请求，如果是get请求则调用render\_index()方法，该方法首先会去寻找源文件目录下analyzer/windows/modules/packages目录下的所有package的名字并且放入集合中，然后会从数据库中查询出所有客户机的名字以及lable等信息 ，最后将这些信息封装好并跳转至submission/index.html页面。当用户点击提交按钮后也会调用submission/views.py文件中的index（）方法，不过是以post方式传递过来的，此时该方法会从表单中获取用户传递过来的信息（包括：包名，超时时间，优先级，客户机类型，强制结束任务时间，url等信息）然后将这些信息封装好，插入数据库中并记录返回的任务ID最后跳转至submission/complete.html页面如图5.5所示。

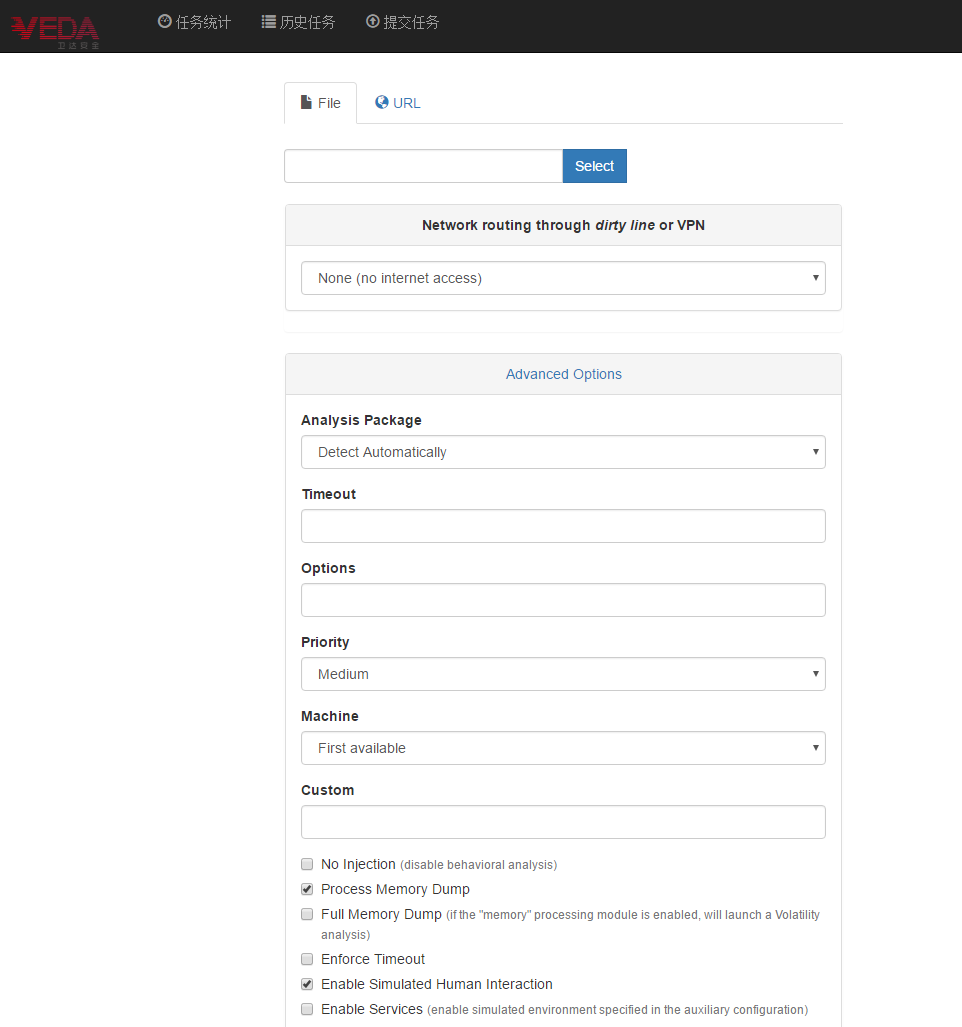


图5.5提交任务界面截图

## 5.4数据库实现

分布式调度模块对数据库的操作采用Python SQLAlchemy 来实现。首先需要在配置文件setting.py中设置数据库的url为SQLALCHEMY\_DATABASE\_URI = "postgresql://dbuser:toor@127.0.0.1/exampledb2"。在distribute/db.py文件中导入Alchemy模块，新建Alchemy对象，然后需要定义三张表对应的类：

定义task类继承自Alchemy.module，它的属性有：id, path, filename, package, timeout, priority, options, machine, platform, tags, custom, owner, memory, clock, enforce\_timeout, node\_id, task\_id, status, submitted, delegated, started, completed, mdd5, category并定义构造方法对他进行初始化。

定义node类继承自Alchemy.module，它的属性有：id,name,url, mode,enabled,mchines并定义构造方法对他进行初始化。

定义machine类继承自Alchemy.module，它的属性有：id,name, platform, tags, node\_id并定义构造方法对他进行初始化。

因为Alchemy框架在服务启动后会自动在数据库创建这些类对应的表，因此并不需要手动去在数据库中创建它们。

数据库创建之后如图5.6所示：

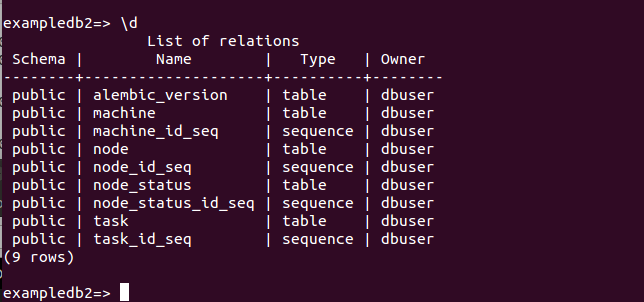


图5.5 数据库截图

## 5.5 本章小结

本章主要介绍了分布式恶意代码检测系统的实现方法，首先介绍了主节点restful api具体是如何实现的，然后介绍了从节点restful api是如何实现的，最后介绍了该系统是如何对数据库进行操作的。

# 6 分布式恶意代码检测系统的测试

## 6.1系统测试原则

系统测试是对系统的组装和验证的测试。系统测试是整个产品系统的测试。目的是验证系统是否满足要求规范，并确定其与要求不一致或矛盾的位置，以便提出更完善的程序。系统测试发现问题后，尝试找出错误的原因和位置，然后进行更正。这是基于黑盒类测试的整体系统要求，应该覆盖系统的所有组件。测试对象不仅包括要测试的软件，还包括软件依赖的硬件，外围设备甚至包括某些数据，一些支持软件及其接口。

## 6.2系统功能测试

系统功能测试用例如表6.1所示：

表6.1 测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 功能模块 | 测试用例 | 测试结果 |
| 1 | Restful api任务提交 | 用例： curl http://localhost:9003/api/task -F file=@calc.exe | 测试结果：成功添加任务 |
| 2 | Restful api节点注册 | 用例：curl http://localhost:9003/api/node -F name=cuckoo1 -F url=http://192.168.16.102:8090/ | 测试结果：成功注册节点 |
| 3 | Restful api节点删除 | 用例： curl -XDELETE http://localhost:9003/api/node/cuckoo1 | 测试结果：成功删除该节点 |
| 4 | Restful api获取节点信息 | 用例：curl http://localhost:9003/api/node/cuckoo1 | 测试结果：成功获取该节点信息 |
| 5 | Restful api获取任务状态 | 用例：curl http://localhost:9003/api/task/1 | 测试结果：成功获取id对应的任务状态 |
| 6 | 从节点Restful api创建任务（文件） | 用例：curl http://192.168.16.102:8090/ tasks/create/file  data = dict(  tid=task["id"],  package=task["package"],  timeout=task["timeout"],  priority=task["priority"],  options=task["options"],  machine=task["machine"],  platform=task["platform"],  tags=task["tags"],  custom=task["custom"],  owner=task["owner"],  memory=task["memory"],  clock=task["clock"],  enforce\_timeout=task["enforce\_timeout"],  ) | 测试结果：成功在此节点创建任务（文件） |

续表6.1

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | 从节点Restful api提供此节点任务列表 | 用例：curl http://192.168.16.102:8090/ tasks/ tasks/ list  params=dict(status=status) | 测试结果：成功获取从节点任务列表 |
| 8 | 从节点Restful api提供此节点某任务物状态 | 用例：curl http://192.168.16.102:8090/tasks/view/ <int:task\_id> | 测试结果：成功获取从节点对应id的任务状态 |
| 9 | 从节点Restful api提供任务报告 | 用例：curl http://192.168.16.102:8090 / tasks/ report/<int:task\_id> | 测试结果：成功获取此节点上对应id的任务报告 |
| 10 | 从节点Restful api提供任务的pcap包 | 用例：curl http://192.168.16.102:8090/ pcap/ get/<int:task\_id> | 测试结果：成功获取此节点上对应id任务的pcap包 |
| 11 | 从节点Restful api提供此节点的状态 | 用例：curl  http://192.168.16.102:8090//cuckoo/status | 测试结果：成功获取此节点的当前状态 |
| 12 | 从节点Restful api提供此节点的客户机信息 | 用例：curl http://192.168.16.102:8090//machines/list | 测试结果：成功获取此节点上所有的客户机信息 |
| 13 | 从节点Restful api创建任务（url） | 用例：curl http://192.168.16.102:8090/ tasks/create/url  data = dict(  tid=task["id"],  package=task["package"],  timeout=task["timeout"],  priority=task["priority"],  options=task["options"],  machine=task["machine"],  platform=task["platform"],  tags=task["tags"],  custom=task["custom"],  owner=task["owner"],  memory=task["memory"],  clock=task["clock"],  enforce\_timeout=task["enforce\_timeout"],  url=task["path"],) | 测试结果：成功获取此在节点上创建url任务 |

续表6.1

|  |  |  |  |
| --- | --- | --- | --- |
| 14 | Web界面：任务统计 | 用例：点击链接  http://127.0.0.1:8765/dashboard/ | 测试结果：成功查看任务统计结果 |
| 15 | Web界面：历史任务 | 用例：点击链接  http://127.0.0.1:8765/analysis/ | 测试结果：成功查看所有任务的列表 |
| 16 | Web界面：提交任务（文件） | 用例：点击链接  http://127.0.0.1:8765/submit/  选择file标签，选择文件，提交 | 测试结果：成功创建任务（文件） |
| 17 | Web界面：提交任务（url） | 用例：点击链接  http://127.0.0.1:8765/submit/  选择url标签，填写url，提交 | 测试结果：成功创建任务（url） |
| 18 | Web界面：查看任务 | 用例：点击链接  http://127.0.0.1:8765/analysis/101/ | 测试结果：成功查看id为101的任务的详细报告 |

## 6.3本章小结

本章主要介绍了分布式恶意代码监测系统的检测，首先描述了系统测试的原则，然后分别对主节点的restful api，从节点的restful api以及web界面的功能测试进行了描述。

# 结论

随着社会的发展和进步，我们越来越离不开互联网，但是恶意软件的数量也越来越多，因此急需一种分布式的能够短时间检测大量恶意软件的系统，本系统的的设计就是为了解决这种需求。

本系统在单机版恶意代码检测系统的基础上加以改造，使用python django框架来实现web界面，使用flask框架来实现主节点和从节点的restful api，数据库操作采用Python SQLAlchemy框架来实现，基本上达到了预期的目标。

但是该系统还存在很多缺陷，比如对获取到的分析结果只能进行查看，不能进行分析，不能建立学习集来判断恶意软件的类型。随着本人能力的提高，之后将继续完善该系统，优化各个模块，增加该系统的功能。

# 参考文献

1. Coulouris. Distributed Systems Concepts and Design.Addison-Wesley,2012
2. 彭渊.大规模分布式系统架构与设计实战.机械工业出版社,2014
3. Andrew S.Tanenbaum, Maartenvan Steen.分布式系统原理与范型.[清华大学出版社](http://baike.baidu.com/item/æ¸åå¤§å­¦åºçç¤¾" \t "_blank),2008
4. 锻钢.加密与解密.电子工业出版社,2008
5. 戚利.Windows PE权威指南.机械工业出版社,2011
6. 罗云彬.Windows环境下32位汇编语言程序设计.电子工业出版社,2008
7. MagnusLieHetland.Beginning Python From Novice to Professional.人民邮电出版社,2010
8. Miguel Grinberg.Flask Web Development: Developing Web Applications with Python.人民邮电出版社,2014
9. Jeff Forcier.Python Web Development with Django.机械工业出版社,2009
10. 霍多罗夫,(美) 迪洛尔夫.MongoDB权威指南.人民邮电出版社,2011
11. 美 Kyle Banker.MongoDB in Action.人民邮电出版社,2014
12. Symantec Corporation. Global Internet Security Threat Report. Symantec Global Internet Security Threat Report Volume XIV,2009
13. Michael Sikorski, Andrew Honig.Practical Malware Analysis the Hands-On Guide to Dissecting Malicious Software.电子工业出版社,2014
14. [闫军伟](http://xueshu.baidu.com/s?wd=authoruri%3A%2854f96fcc66f4fee3%29%20author%3A%28%E9%97%AB%E5%86%9B%E4%BC%9F%29%20%E5%9B%BD%E9%98%B2%E7%A7%91%E5%AD%A6%E6%8A%80%E6%9C%AF%E5%A4%A7%E5%AD%A6%E8%AE%A1%E7%AE%97%E6%9C%BA%E5%AD%A6%E9%99%A2&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson&sort=sc_cited" \t "_blank),[钟求喜](http://xueshu.baidu.com/usercenter/data/author?cmd=authoruri&wd=authoruri%3A%28e0be28c4664e3c91%29%20author%3A%28%E9%92%9F%E6%B1%82%E5%96%9C%29%20%E5%9B%BD%E9%98%B2%E7%A7%91%E6%8A%80%E5%A4%A7%E5%AD%A6%E8%AE%A1%E7%AE%97%E6%9C%BA%E5%AD%A6%E9%99%A2" \t "_blank),[贾欣](http://xueshu.baidu.com/s?wd=authoruri%3A%28f5cc154593ef0a02%29%20author%3A%28%E8%B4%BE%E6%AC%A3%29%20%E6%AD%A6%E8%AD%A6%E7%9F%B3%E5%AE%B6%E5%BA%84%E6%8C%87%E6%8C%A5%E5%AD%A6%E9%99%A2%E4%BF%9D%E5%AF%86%E6%A1%A3%E6%A1%88%E4%B8%9A%E5%8A%A1%E6%95%99%E7%A0%94%E5%AE%A4&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson&sort=sc_cited" \t "_blank),[王茜](http://xueshu.baidu.com/s?wd=author%3A%28%E7%8E%8B%E8%8C%9C%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank).[基于行为的分布式恶意代码检测技术](http://xueshu.baidu.com/s?wd=paperuri%3A%283487d7402287e55d8d0566bd755e5da6%29&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http%3A%2F%2Fkns.cnki.net%2FKCMS%2Fdetail%2Fdetail.aspx%3Ffilename%3Djyxh201109036%26dbname%3DCJFD%26dbcode%3DCJFQ&ie=utf-8&sc_us=8785961567256780944" \t "_blank),2011(9):126~12
15. [韩晓光](http://yuanjian.cnki.com.cn/Search/Result?author=%E9%9F%A9%E6%99%93%E5%85%89" \t "_blank).恶意代码检测关键技术研究.北京科技大学,2015

致谢

本篇论文是在庞海波老师的指导下写完的，大学四年来庞老师无论是在学习工作还是在生活上都给予我无微不至的关怀。庞老师对待学术勤奋，进取，对待同学严厉，使我深受启发。我在毕业之后的路还很长，毕业不是结束而是一个开始。庞老师尽管工作非常忙，但仍然占用休息时间为我指导，给予我帮助。在这里我要向庞老师表达我最真挚的感谢。

回首大学四年，感慨良多。感谢班里的各位同学对我的照顾和帮助，使我的能力的到了提高。感谢公司各位同事对我的帮助，使我在软件安全领域收获良多。

谨以此文献给所有关心帮助过我的各位同事，同学和朋友们。

赵晓宁2017年5月20日于郑州大学