## 摘  要

当前互联网的网络空间由最初的混沌到现在快速扩展，互联网设备的也运用越来越广泛，全世界上网设备的激增导致连IPV4地址都无法包含下去，可以想象在未来，会有越来越多设备在互联网中相连接，以互联网为基础上的延伸和扩展，人类生活的延伸扩展到了一切物品之间的信息交换与通信。但是越来越信息化的时代也伴随着越来越多的风险，通过网络空间搜索引擎，我们可以快速的知道，全世界大约有多少网络摄像头，有多少服务器，某个开源组件的使用情况等等。网络空间搜索引擎在网络空间测绘，态势感知，企业资产安全管理等网络安全领域发挥着巨大的作用。

阿尔文 托夫勒在《第三次浪潮》中提到，“谁掌握了信息，控制了网络，谁就将拥有整个世界”。就“棱镜”事件来看，整个中国社会必须重新认识到网络空间在整个国家安全体系下的重大意义。本文设计的目标便是制造这样一个系统，用于监控整个网络设备，甚至得到某个地区的设备的相关分布情况。

关键词：网络空间；网络资产；搜索引擎；监控

## ABSTRACT

The current Internet space of the Internet has expanded rapidly from the initial chaos to the present, and the use of Internet devices has become more widespread. The surge in Internet devices worldwide has prevented even IPV4 addresses from being included. It is conceivable that there will be more and more in the future. Devices are connected in the Internet, extended and extended on the Internet, and the extension of human life extends to the exchange and communication of information between all items. But the era of more and more information is accompanied by more and more risks. Through the cyberspace search engine, we can quickly know how many web cameras, how many servers, the use of an open source component, etc. Wait. The cyberspace search engine plays a huge role in the field of network security such as network space mapping, situational awareness, and enterprise asset security management.

Alvin Toffler mentioned in the "Third Wave" that "whoever has the information and controls the network will have the whole world." As far as the "prism" incident is concerned, the entire Chinese society must re-recognize the significance of cyberspace under the entire national security system. The goal of this paper is to create a system for monitoring the entire network equipment and even the relevant distribution of equipment in a certain area.

**Keywords:** Cyberspace; Network assets; Search engine; Monitori

目  录

1 前言 1

1.1 研究目的 1

1.2 发展概况 1

1.3 开发环境 1

1.4 其他环境 2

1.4.1 Django开发框架 2

1.4.2 Redis缓存服务器 2

1.4.3 Elasticsearch搜索解决方案 2

1.4.4 Nmap网络探测工具 2

1.4.5 Masscan网络扫描工具 3

1.4.6 Docker容器化解决方案 3

2 系统分析 4

2.1 主要功能 4

2.2 Client端设计原理 4

2.2.1 网络服务识别原理 4

2.2.2 网络端口快速扫描原理 4

2.2.3 WEB指纹识别原理 5

2.2.4 插件化扫描概述 5

2.2.5 去重、缓存调度算法 5

2.3 WEB端设计原理 6

2.3.1 Restful接收数据 6

2.3.2 大数据搜索解决方案 6

2.3.3 资产关联性算法 6

2.4 扫描端与WEB端的数据交互 7

2.4.1 MQ的选择 7

2.4.2 Redis消息队列的实现 7

2.4.3 相关操作命令 7

2.4.4 代码原型 7

2.5 扫描端节点监控 8

2.5.1 节点监控思路概述 8

2.5.2 节点注册 8

2.5.3 节点异常条件 8

2.5.4 节点与WEB的通信 8

2.5.5 节点销毁 8

2.5.6 WEB端获取节点信息 9

3 扫描端设计 10

3.1 需求分析 10

3.1.1 功能需求 10

3.1.2 性能需求 10

3.2系统总体架构设计 10

3.3系统功能的插件化设计 11

3.3.1 Python代码原型 12

3.4调用外部仓库插件 13

3.5IP地理位置查询 13

3.6收集器设计 14

4WEB端设计 15

4.1设计模式MVC架构 15

4.2WEB端总体设计 15

4.2.1 前端布局设计 15

4.2.2 API模块设计 18

4.2.3 资产模块设计 18

4.2.4 搜索模块设计 19

4.3数据存储结构 19

4.3.1 域名信息存储结构 19

4.3.2 IP信息存储结构 20

5运行测试 22

5.1WEB端运行测试 22

5.1.1 首页 22

5.1.2 域名资产信息 23

5.1.3 系统统计页 24

5.1.4 资产详情 25

5.2扫描端运行测试 27

6自动化部署 28

6.1使用docker目的 28

6.2WEB Dockerfile 28

6.3扫描端dockerfile 28

6.4多个镜像启动 28

7总结与展望 30

7.1开源 30

参考文献 31

1 前言

1.1 研究目的

网络安全是国家的重点发展目标之一，在防卫自身安全的同时，我们需要了解某个国家的大概网络基础设施情况，有多少的使用率，这样才能在未来的网络空间战中从容不迫，有的放矢。

不仅是了解网络基础设施，当某个使用量大的服务组件被爆出存在漏洞时，我们也需要这样一个系统，来了解使用了该服务组件的主机、相关等等，做好预防措施。当公司需要管理其资产时，只需要在资产配置上输入自己企业的相关域名，ip地址，该引擎就会自动匹配与其相应的服务器、域名等信息，同时也包含在该服务器上运行的各项服务，WEB使用的开源CMS类型，让你一眼就能够了解自己企业相关状况。

1.2 发展概况

在2009年进行的黑客大会DEFCON会上，一位名叫约翰 马瑟利的黑客发布了一款名为“Shodan”的搜索引擎。Shodan 通过扫描全网设备并抓取解析各个设备返回的 banner 信息，通过了解这些信息 Shodan 就能得知网络中哪一种 Web 服务器是最受欢迎的，或是网络中到底存在多少可匿名登录的 FTP 服务器。

国内这些年也推出了它们的跟随者，Zoomeye和FOFA，功能方面大体都相近，各自都有网络指纹识别，服务识别，组件查找，资产发现等功能。

1.3 开发环境

开发环境在Mac OS Mojave，版本为10.14.2，在实际部署的时候WEB端和Client端代码都部署在Linux机器上。

基于程序的灵活性和编写的方便，选择使用python3作为主要的编程语言，在WEB后端方面，python也有成熟的框架Django提供使用。

Python是一种计算机程序设计语言。是一种动态的、面向对象的脚本语言，最初被设计用于编写自动化脚本(shell)，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。

1.4 其他环境

现代软件科学有很多成熟框架的框架，我们只需要站在巨人的肩膀上享受这一切。

1.4.1 Django开发框架

Django是一个开放源代码的Web应用框架，由Python写成。采用了MVT的软件设计模式，即模型Model，视图View和模板Template。它最初是被开发来用于管理劳伦斯出版集团旗下的一些以新闻内容为主的网站的。并于2005年7月在BSD许可证下发布。这套框架是以比利时的吉普赛爵士吉他手Django Reinhardt来命名的。

Django的主要目标是使得开发复杂的、数据库驱动的网站变得简单。Django注重组件的重用性和“可插拔性”，敏捷开发和DRY法则（Don't Repeat Yourself）。在Django中Python被普遍使用，甚至包括配置文件和数据模型。

1.4.2 Redis缓存服务器

Redis是一个开源的，支持网络，基于内存的键值对存储系统。高可用高并发的支持使得它成为了最流行的键值对存储数据库。

1.4.3 Elasticsearch搜索解决方案

Elasticsearch 是一个分布式、RESTful 风格的搜索和数据分析引擎，能够解决不断涌现出的各种用例。作为 Elastic Stack 的核心，它集中存储您的数据，帮助您发现意料之中以及意料之外的情况。

开源的 Elasticsearch （以下简称 Elastic）是目前全文搜索引擎的首选。

它可以快速地储存、搜索和分析海量数据。维基百科、Stack Overflow、Github 都采用它。

1.4.4 Nmap网络探测工具

Nmap (“Network Mapper(网络映射器)”) 是一款开放源代码的 网络探测和安全审核的工具。它的设计目标是快速地扫描大型网络，当然用它扫描单个 主机也没有问题。Nmap以新颖的方式使用原始IP报文来发现网络上有哪些主机，那些 主机提供什么服务(应用程序名和版本)，那些服务运行在什么操作系统(包括版本信息)， 它们使用什么类型的报文过滤器/防火墙，以及一堆其它功能。虽然Nmap通常用于安全审核， 许多系统管理员和网络管理员也用它来做一些日常的工作，比如查看整个网络的信息， 管理服务升级计划，以及监视主机和服务的运行。

1.4.5 Masscan网络扫描工具

Masscan是最快的互联网端口扫描工具，它最快可以在六分钟内扫描全球网络，高达1000万/秒的并发。它更像scanrand，unicornscan以及使用异步传输的ZMap。主要区别在于它比其他扫描器更快。此外，它更灵活，允许任意地址范围和端口范围。

1.4.6 Docker容器化解决方案

Docker使用Google公司推出的Go语言编写，基于Linux隔离内核实现，属于操作系统层面的虚拟化技术。由于隔离的进程独立于宿主和其它的隔离的进程，因此也称其为容器。Docker 在容器的基础上，进行了进一步的封装，从文件系统、网络互联到进程隔离等等，极大的简化了容器的创建和维护。使得 Docker 技术比虚拟机技术更为轻便、快捷。

2 系统分析

2.1 主要功能

本论文主要是设计一款资产发现引擎，包括Client端和WEB端，Client端用于不断扫描并识别网络的服务，即本文实现的“网络发现引擎”，本文统一使用“CLient”端称呼它。WEB端用于接收Client端返回的结果，并以人性化友好的方式展现出来。在设计中需要考虑非关系型数据如何存储，如何快速搜索相关数据，以及Client端如何快速的扫描和识别网络，针对一个域名，如何识别其所使用的技术等等。

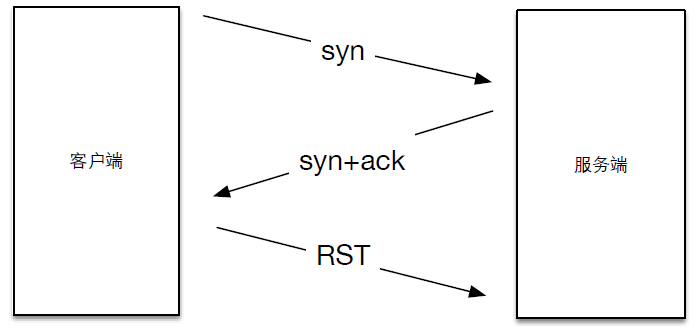
2.2 Client端设计原理

2.2.1 网络服务识别原理

TCP协议中三次握手可判断一个端口的存活，而一个服务器最多可开启65535个端口，在TCP的最后一次握手中，服务器会返回一段信息给请求者。所以可以使用TCP连接对一个目标ip进行65535次连接，在连接成功的端口中，获取到服务器最后返回的一段信息，由于不同的服务ftp，http会有不同的返回结果，根据这些结果作为指纹，来识别出该端口所对应的服务。

2.2.2 网络端口快速扫描原理

TCP是可靠的面向连接的协议，如图1.1所示，一个完整的TCP会话每个过程都有不同的状态。快速的端口扫描器原理都是基于无状态的扫描。在确认端口打开后，通过RST放弃建立连接。



图**1.1** Tcp连接图

Masscan就是这样一种扫描器，可以扫描互联网规模的端口扫描器，可用于互联网或内部网络的大规模调查。它可以选择高达2500万个数据包/秒的扫描速度，这个速度足以在3分钟内扫描互联网。本文所设计的引擎也是通过调用masscan来进行扫描。

2.2.3 WEB指纹识别原理

开源的WEB系统，在静态文件(例如html,js,css)中会包含一些特征字符串，即使没有，固定url文件也是一种特征字符。例如wordpress会在robots.txt中会包含wp-admin之类，默认样式会在首页中包含generator=wordpress xx，页面中会包含wp-content路径等等，以此类推，几乎所有开源CMS都有类似的指纹特征。

识别WEB指纹的方式，通过获取一段特定的路径，判断该路径下的返回文本（通过md5或者正则表达式），在指纹库中存在相对应的指纹，即可判别该URL所属于的WEB应用。

2.2.4 插件化扫描概述

插件化有助于程序的扩展，得益于python天生就适合插件化，在本论文的设计思想中，插件化的思想是整个程序设计中的重点，在进行WEB扫描的时候，所有的扫描功能（敏感文件扫描，web指纹识别等等）都将作为插件，而“调度控制中心”只需要在收集好URL和IP之后动态的调用插件即可。

2.2.5 去重、缓存调度算法

当一个URL或IP已经被扫描过了，但是在另一次扫描中重复了这些URL或IP，为了节省扫描资源，这些重复的URL和IP应该被舍弃，而不应该加入到扫描中。又或者，我只需要扫描的这段URL和IP存在一段时间，这一段时间之后，如果进行的新的扫描包含了这些URL和IP，我们需要让他加入到扫描。

这其实就是缓存数据库的概念，我们只需要维护一个“已经扫描”的数据库，每进行一次扫描的时候，只需要查询这条记录是否在数据库中存在，若存在则不进行扫描，而且我们要求数据库中的记录只需要缓存“一段时间”，这一段时间过，数据库会自动删除这些数据。而且我们也要求数据库的读取速度和性能尽可能的快。

缓存数据库可以使用Redis。Redis是一种内存存储数据库，可以作为数据库，高速缓存和消息队列等服务，刚好满足我们的要求。基于Redis，就可以简单的设计一个程序来满足我们的要求。

2.3 WEB端设计原理

2.3.1 Restful接收数据

Restful framework是目前最流行的 API 设计规范，用于 Web 数据接口的设计。在Client端传入数据到WEB端的时候，采用RESTful架构接收，可以很方便的从URL中得到表达意义（具体展现在URL的表现层状态转化），而且在设计结构的时候会显得非常清晰，容易理解。

2.3.2 大数据搜索解决方案

因为在设计中扫描的是全球网络，在此基础上分析出来的各种服务、指纹、应用、版本也都需要存入数据库。所以在搜索方案上面，我们需要考虑一是由于数据量巨大，一些常见的数据库搜索可能不满足我们的在时间上的需要，二是我们需要寻找一款非关系型的数据来存储这些数据。在衡量了各项数据指标后，开源的Elasticsearch是我们的解决方案。

ElasticSearch是一个搜索服务器。它提供了全文搜索引擎和基于RESTful web接口。有着很强大的数据分析和处理功能，对于中文支持也比较友好，可以安装中文分词插件对中文进行分析。

ElasticSearch在有数据入库时都会对其进行分词并索引，大大增强了查询速度，独有的结构化查询语言，可以很方便的找到想要的数据以及对应数据的相关信息。

2.3.3 资产关联性算法

在本论文的设计中，我们需要设计一个算法，在存储数据的数据库中寻找我们需要的“某地”的资产相关情况。例如，我们需要查看“百度”旗下的相关资产信息，而WEB程序并不可能自动的通过输入的“百度”关键词来找到所有相关百度的资产。

在本论文设计中，WEB后端有一个资产发现的配置页面，需要我们手动输入“百度”相关的资产，列如域名，\*.baidu.com、\*.hao123.com，ip资产等等，程序通过这些配置来寻找相关的资产情况。Elasticsearch提供了通配符wildcard的搜索方式正好可以满足我们的要求。通过Elasticsearch添加bool查询条件，并使用wildcard的查询方式即可满足查询需求。但是此查询在效率上稍低。

2.4 扫描端与WEB端的数据交互

在扫描端架构和WEB端架构都完成时，接下来考虑的是从WEB端下发任务，如何让Client端执行任务。在分布式架构中有一个术语叫做“Message Queue”即使用消息队列，本系统也采用这样的方式。消息队列是消费-生产者模型的一个典型的代表，一端往消息队列中连续写入消息，而另一端则可以读取或者订阅队列中的消息。本论文采用的架构中，WEB端作为生产者将需要处理的数据写入到消息队列，扫描端作为消费者从消息队列中读取数据并进行处理。

2.4.1 MQ的选择

较为成熟的MQ产品有IBM WEBSPHERE MQ、rabbitmq等等

因为本系统的需求比较简单，不需要复杂的状态管理，同时也为了避免使用的服务太多不好维护，选择使用redis数据库来作为消息传输队列。redis服务还作为扫描端去重的服务，正好也是一物两用。

2.4.2 Redis消息队列的实现

redis支持列表操作，redis的列表是一个简单的双端列表，你可以选择从左边插入数据还是从右边插入数据，相应的，也可以选择从左边读取数据还是从右边读取数据。在本系统的实现中，只要约定从一端插入，从一端取出即实现了消息队列的功能。

redis本身就是线程安全的，所以无需考虑在插入的过程中读取操作如何进行。

2.4.3 相关操作命令

•lpush key value 插入value到key左端

•lpop key 移除并获取第一个元素，如果此时列表为空则返回None

•blpop key 移除并获取第一个元素，如果此时列表为空则一直等待

2.4.4 代码原型

redis消息队列的Python代码表达

importredis# 导入redis模块，通过python操作redis 也可以直接在redis主机的服务端操作缓存数据库

pool=redis.ConnectionPool(host='127.0.0.1', port='6379',

decode\_responses=True)# host是redis主机，redis默认端口是6379，解码返回的消息为真

redis\_con=redis.Redis(connection\_pool=pool)

redis\_con.lpush("w12scan\_scanned", ip)

2.5 扫描端节点监控

在实际运行过程中，由于扫描节点的增多，管理者需要查看各扫描节点的运行状态是否正常，所以设计一套能在WEB端直接方便查看节点状态的功能是必要的。节点监控不仅能查看各节点运行状态，（包括正在运行的任务，运行成功的任务，节点是否正常，以及节点记录的日志等等）还能处理失效的节点，查看原因，删除节点等。

2.5.1 节点监控思路概述

节点监控的采用的是扫描端向redis每隔5分钟(5\*50s)写一次数据，内容为当前时间，WEB端读取当前时间-redis读取的时间，若大于5分钟则报告节点异常。

2.5.2 节点注册

扫描端运行时向redis数据库写入特定的key用以注册节点，以"w12node"开头，后面为自定义的节点名称，同时向该节点注册三个字段。

•last\_time 用于保存节点最后写入时间

•running 用于保存正在运行的任务数量

•finished 用于保存已经完成的任务数量

2.5.3 节点异常条件

节点异常在WEB端判断，若WEB端当前时间-redis读取时间>5min则提醒节点异常，显示删除节点按钮。

2.5.4 节点与WEB的通信

•节点向指定的redis队列中写入消息 如w12\_log\_[nodename]

•节点向指定redis key中写入消息，如w12\_node\_[nodename]

2.5.5 节点销毁

节点销毁的选项是在节点已经无法通信的情况下进行的，此时只需要删除redis数据库中的两个keyw12\_log\_[nodename]和w12\_node\_[nodename]即可。若在节点能够通信的情况下销毁，还需要保证联通节点一起销毁。

2.5.6 WEB端获取节点信息

Redis可以通过模式匹配如w12\_node\_\*的方式来匹配到所有节点，以此遍历数据来读取每个节点的信息，然后通过读取w12\_log\_[nodename]读取扫描节点的日志log。

3 扫描端设计

3.1 需求分析

在整个扫描端的设计中，需要弄清楚两种需求的分析，并证明其有效性与可行性。针对《网络资产发现引擎》这个大框架的扫描端来说，需求分析尤为重要，它决定了在未来《网络资产发现引擎》得到数据的有效性与真实性。需求分析主要从功能需求与性能需求展开来说，设计的原则是在尽可能满足功能需求的前提下尽量满足性能需求。

3.1.1 功能需求

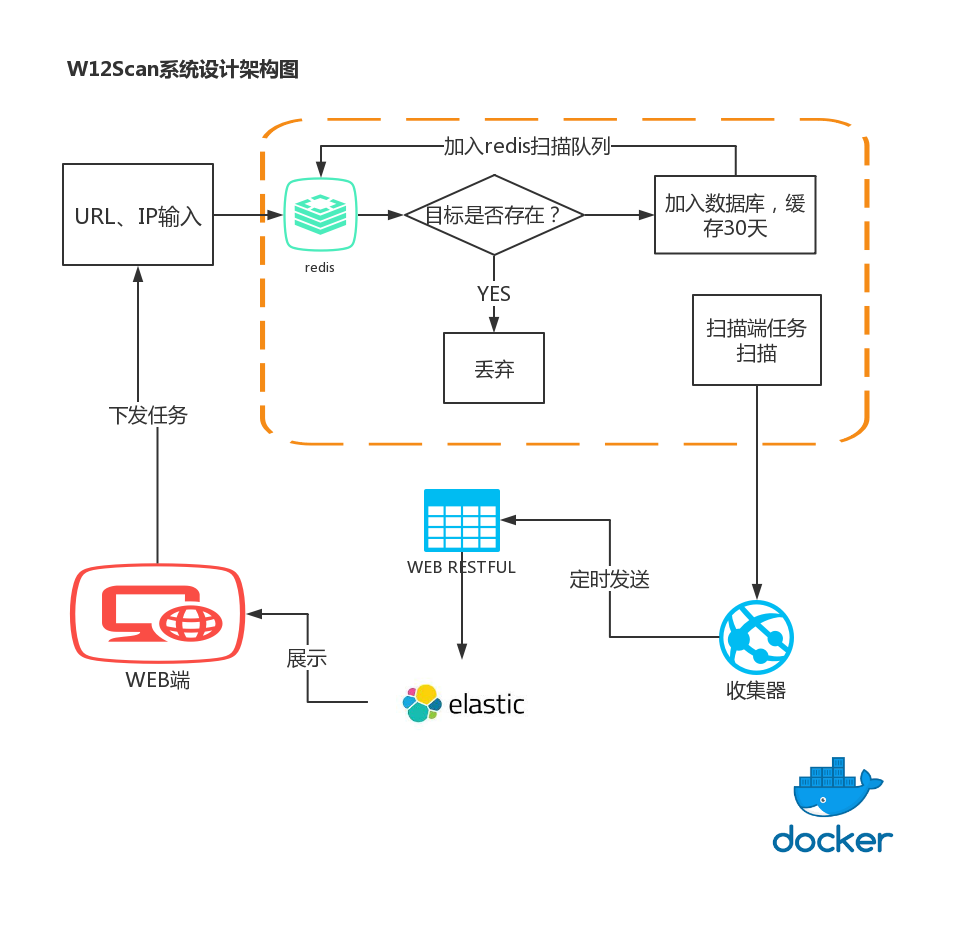
功能上的需求要求该扫描端引擎能够接受大量的域名或ip地址，扫描端引擎对每个域名和ip地址进行访问请求并进行“分析”，得到WEB特征指纹并向WEB端的Web restful接口发送结果。“分析”是指，通过一个IP或者域名，尽可能得到它的其他详细信息，例如对于一个ip地址来说，得到它的来源地、开放端口、每个端口所运行的服务，对于一个域名来说，需要得出它使用的服务器，操作系统，运行的脚本环境，网站的标题，网站使用的相关技术，使用了哪些开源的组件服务等等。

3.1.2 性能需求

针对大量的ip地址和域名的扫描，在IO等待上非常消耗时间，为了加快扫描速度，在IO等待的时间内继续扫描其他目标，使用开源的masscan作为端口扫描器工具，在网站域名的扫描方面，使用多线程技术对网站进行探测。 因为masscan结束扫描时会默认等待10s，太耗费时间，所以在扫描端的设计上，接收到的IP地址会先缓存到一个列表中，当缓存的IP数量大于一定数值时，才会调用Masscan进行扫描，此即是IP缓冲池的设计，同理，也会设计域名缓存池，当域名缓冲池中域名大于一定数值后，将缓冲池中的域名发送到多线程框架中进行域名的分析与探测。

3.2 系统总体架构设计

需求分析中已经详尽描述了扫描端的相关需求，现在需要统筹一下WEB端与扫描端的的总体，计划WEB端如何与扫描端的通信方式使用redis作为中间数据库的消息队列，设计图3.1展示了扫描端的运行流程以及如何与WEB进行通信的流程。



图**3.1**  系统总体设计流程图

如图所示，client扫描端在扫描完并取得相应结果后会想web端的restful接口发送数据，之后web端再将数据存入到elasticsearch数据库中。WEB端也提供restful接口来进行添加扫描的目标，扫描目标会首先加入到redis数据库中，client扫描端在从对应的redis数据库中读取内容。

URL、IP的输入会先进行一次“去重过滤算法”，过滤出来的目标进入“任务分发调度器”，任务分发调度器会识别该目标是url还是一个ip地址，根据流程图送入不同的操作流程。最后将结果放入“收集器”当中，“收集器”会自动将结果提交到WEB restful接口当中。

3.3 系统功能的插件化设计

每当“分发调度器”收到一个域名或一个ip，会缓存一定数量，当达到这个数量便会开始进行扫描操作，扫描操作会调动“插件”进行，每个功能模块作为一个插件单独运行。针对域名和ip这两种情况会有两种不同的情况的插件集合。

针对ip服务，会先启用masscan扫描工具过滤出开放的ip端口，再调用nmap服务识别每个端口所对应的服务类型，如果发现开放的端口中有http服务，会加入到域名扫描队列中运行，如果有其他的服务，例如redis，mysql等等，会调用相对应的扫描插件进行密码爆破，漏洞攻击等服务。

poc插件：PoC(全称: Proof of Concept), 中文译作概念验证。在安全界，你可以理解成为漏洞验证程序,和一些应用程序相比，PoC 是一段不完整的程序，仅仅是为了证明提出者的观点的一段代码

针对域名服务，先发出一条普通的http请求，得到服务器返回的header和body，针对header和body进行正则匹配和关键词分析，可以从中得到一些服务器信息，插件化的意义在于，每个插件都可以得到这个域名的header和body用作分析，并返回对应结果。然后，会对域名进行一些敏感文件的扫描搜集，例如.git、.svn等敏感文件，一些敏感目录，后台目录等等。最后通过指纹分析得到该域名使用的cms系统名称，通过该名称调用与之相对应的攻击插件。由于这些攻击插件的时效性问题，一个新漏洞爆发，一个攻击利用链会被迅速利用，所对应的poc插件具有时效性，当过了两三个月后，该poc可能就失效了，所以我们需要经常性的维护poc插件集。

poc插件如果添加在client程序里，难免会因为失效而过时，但是python语言的天生灵活性，攻击插件可以从特定的网络上下载，我们只需要在github上维护一个poc插件仓库，扫描器工作流程在识别完对应服务后会自动从该github上获取有关自身信息的插件，进行攻击，从而保证了插件维护的及时性和有效性。

3.3.1 Python代码原型

Python是一门灵活的动态语言，它是边解释边执行的脚本语言，因为这个特性，所以可以在运行的时候动态修改类或变量，当然也可以在动态运行当中加载其他模块，加载方式可以是文件或纯文本。

我们要实现从源码加载程序的功能，这是一个非常hack的事情。顺便可以幻想一下，如果可以直接从运行中的程序中加载源码，获得其功能并再次运行，那么是不是我们只需要实现一个类似“加载器”的东西，其主要功能的代码全部从网上下载？是的这是可以实现的，我们在线调用漏洞程序也是依据此原理进而联想到的。

用Python来实现这个功能，Python的内置函数库中有importlib库用于加载模块的操作，以上在线加载的所有功能依赖该库即可完成。

通过阅读Python官方文档的相关说明，要实现从源码中加载模块的功能，需要我们自己实现一个加载器，在该加载器中声明加载源码的路径，文件名，以及申明一个唯一值来代表该模块。需要注意的是，由于我们是从网络上下载源码，源码存储的地方是内存中，我们根本不需要将从网络上读取的源码单独写到一个文件中再从文件中调用，我们只需要读入到内存中，Python有自带的Compile函数将它编译成模块。同时编译模块的路径我们不需要将它写成真实路径，我们只需要一个唯一的伪路径来表达即可，每个模块的唯一值从该源码文件的md5值计算得出。

3.4 调用外部仓库插件

随着各种漏洞的爆发，poc的更新也非常频繁，如果在设计中将poc插件放置于“云端”，而扫描器只需要访问一个特殊的接口，便可以从云端加载相关插件，既能保证更新速度，也方便维护和使用。

在本文的设计中，选择使用github作为开放的云端，选择它的理由不仅仅是因为它免费，方便好用，更在于维护的这些插件随时可以和全世界的开发者们进行交流，沟通。

我的设计思想是，维护一个github仓库，专门用来存放相关的poc文件，poc文件使用python3统一编写，并且约定相应格式。在github仓库的根目录下存在一个api.json的文件，json格式，内容包括漏洞类型，漏洞对应的软件名称，时间，和路径。根据此，可以找到需要的poc文件，按照相对应的文件路径进行加载。

相关github仓库示例:https://github.com/boy-hack/airbug

3.5 IP地理位置查询

扫描端需要获取ip的地理位置，解决方案是网络上的查询api和开源的地理位置信息库。在测试多次之后发现网络上的查询API虽然精度比较高但是对请求频率有限制，为了无限制的使用决定使用开源的地理位置信息库。最后决定使用网络上开源的GeoLite2（GeoLite2 Country）数据库，也提供了方便的api，而且有python的调用例程。python代码调用代码如下

import geoip2.database

def geoip(arg):

filename=os.path.join(PATHS.DATA\_PATH,"GeoLite2", "GeoLite2-City.mmdb")

reader = geoip2.database.Reader(filename)

response = reader.city(arg)

d = {

"country\_id": response.country.iso\_code,

"country": response.country.name,

"region": response.city.name

}

return d

print(geoip('1.1.1.1'))

3.6 收集器设计

收集器在于统一收集扫描结果，并自动上传给web restful接口进行下一步处理。当一个域名或者ip扫描结束后，会通知收集器“我结束了”，此时收集器会将其结果放入缓存队列当中。在队列数量达到一定数量后，会自动读取队列中的结果，批量发送给web端的restful接口存储数据。

整个搜索发现引擎实际上只处理了两种类型的元数据，即域名类和ip类，这两种类型的结果结果也不相同，所以收集器使用两种缓存队列来分别存储这些结果，在提交给web端时，也是读取这两类队列的结果。

在对域名处理过程中，每个扫描插件都会将扫描结果存入收集器当中，但有时，某个扫描插件可能会需要当前域名的另一个插件所提供的扫描信息，这个时候，就需要为收集器设计换一个接口来返回此域名对应的信息，同时还需要兼顾到插件化扫描的执行顺序，例如顺序执行a、b、c插件时，a插件第一个执行，c插件最后一个执行，此时若a插件需要调用c插件结果的相关信息，必然调用不到，所以设计器在设计当中也需要考虑此问题。

4 WEB端设计

4.1 设计模式MVC架构

MVC模式是WEB服务器领域里著名的开发模式，模式的思想是将WEB应用分为三种层次，模型层（M）、控制器（C）和视图（V）三层。三种模式以松耦合的方式连接在一起，各司其职，模型层负责与业务数据库打交道，控制器用于接收用户请求将它传送给视图，视图层负责与用户进行交互。而使用的开发框架Django默认是MVC的模式架构。

4.2 WEB端总体设计

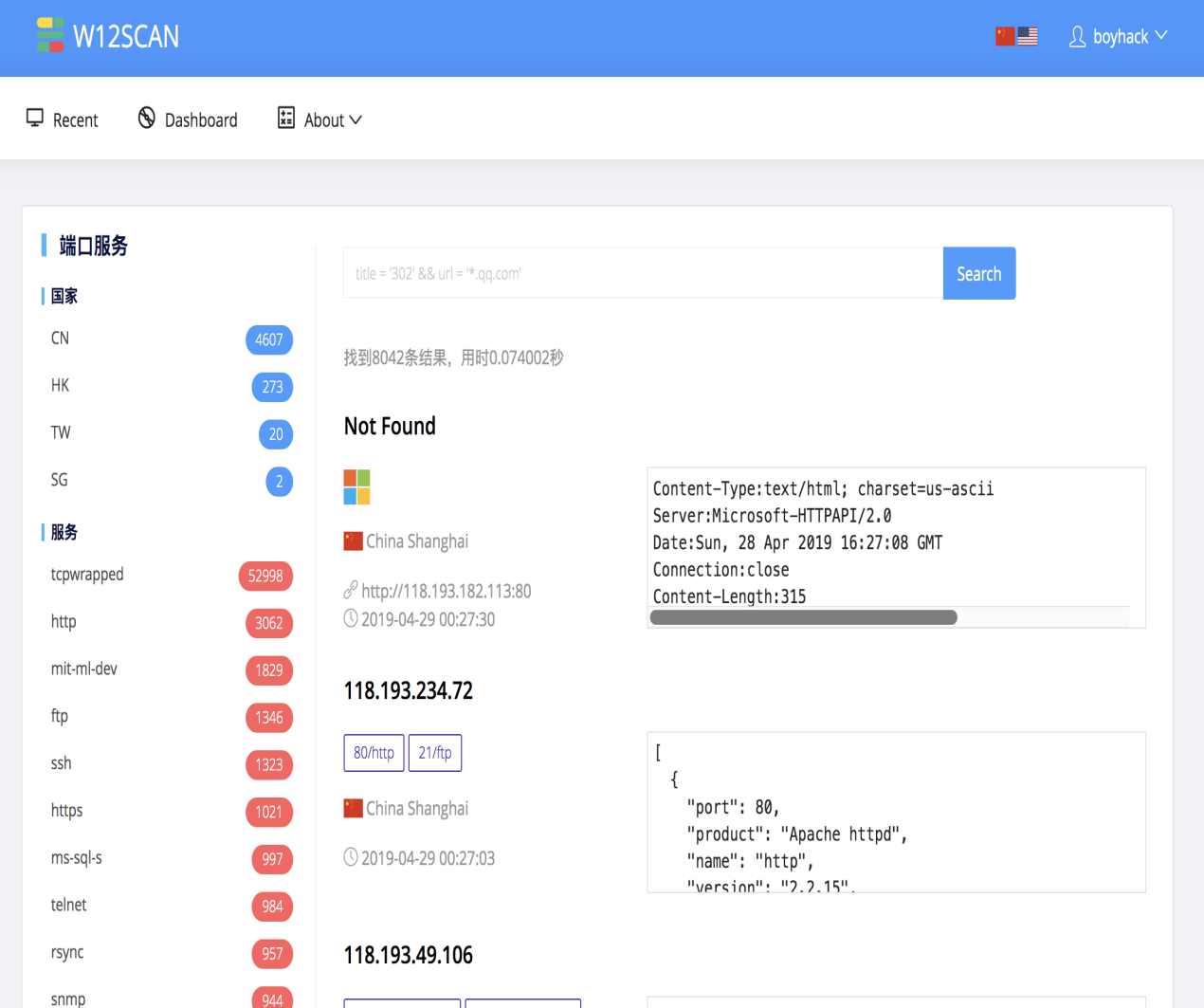
WEB端的作用在于数据显示和与用户交互，前端部分使用Html5+Bootstrap+JavaScript完成,后端使用Python语言和Django框架进行开发，数据库使用MySQL，由于功能需求，需要快速的使用搜索服务，所以还需要使用Elasticsearch这个服务。处于对资产配置功能的需求，使用sqlite3数据库存储，WEB端中“任务队列”还需要使用redis数据库，用作任务的去重以及对扫描端的通信中间数据库。

4.2.1 前端布局设计

前端使用基于bootstrap框架编写的模板，bootstrap已经为我们写好了web前端场景中经常使用到的前端组件，我们只需要将这些组件稍微组合成我们需要的样子即可。本设计的前端主要有三个页面模块需要设计，分别是前端首页的布局，前端导航页布局，前端资产显示页面的布局。前端的布局主要是上下布局，上为导航栏的一些导航链接，下为展示界面。首页的布局主要以蓝色为主色调，其他颜色为作为辅助。

4.2.1.1 首页布局

因为本设计的灵感来源是国内的zoomeye和国外的shadon，所以首页布局的灵感也是模仿自它们。最上面是一个导航栏，显示跳转几个主页面的路由，左侧是数据的统计信息，用于统计“国家”、“服务名称”、“开放端口”、“相关组件”等等。右边显示的是通过左边的筛选方式过滤选择出的相关主机以及IP地址，分别展示它们的基本信息，网页标题、归属地，网页body信息、相关链接、加入时间等等，还有比较细节部分的，网页所使用的一些技术也会用其对应的logo图标显示，如果检测到存在漏洞，也会在标题右侧以红色方框提醒用户该网站可能会存在威胁等。图4.1展示了详细布局。



图**4.1**  首页布局图

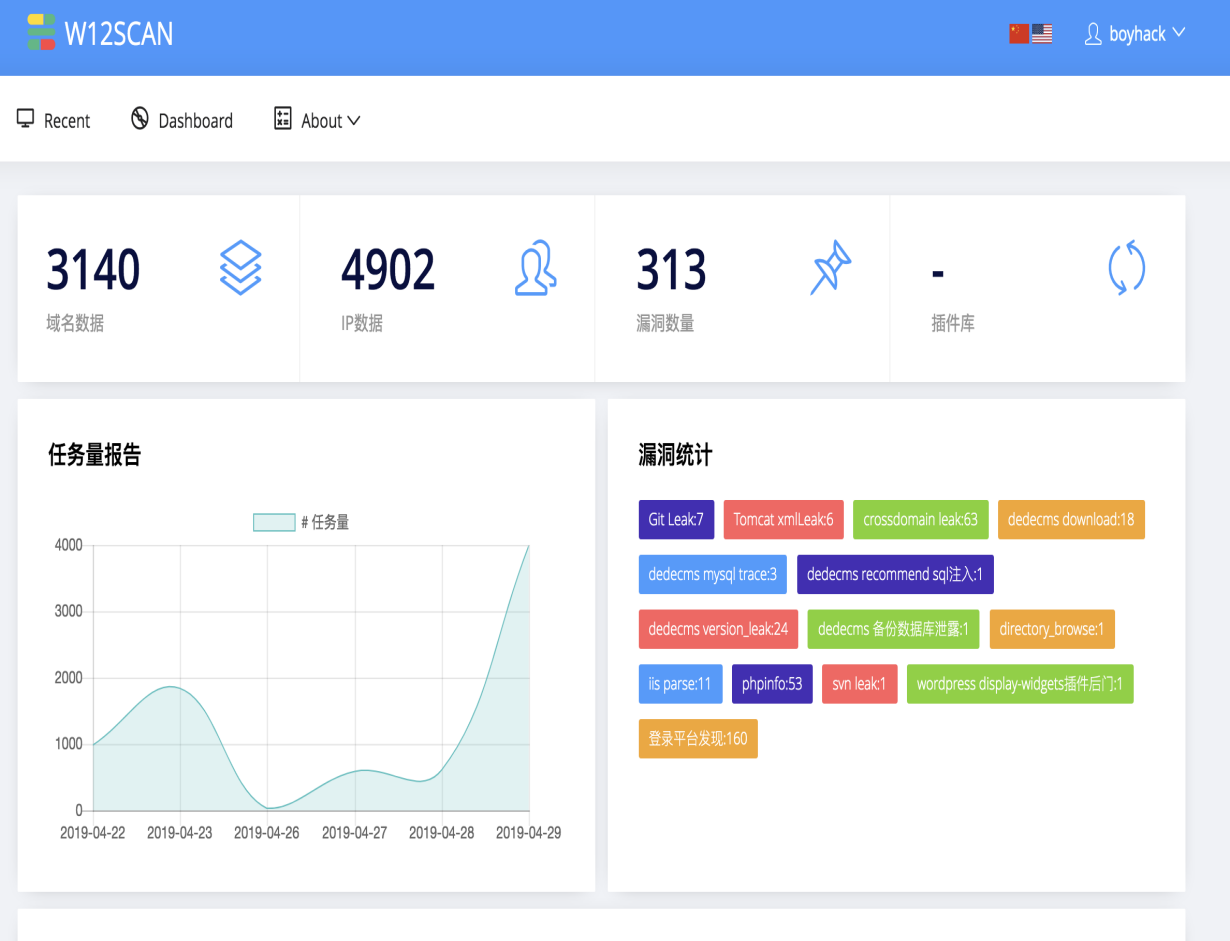
4.2.1.2 导航布局

Dashboard主要用于直观显示扫描端系统的状态信息，统计域名、IP、漏洞数量的等数据。通过图表展示扫描端扫描的任务量，节点的运行状态等信息。

扫描端在设计时是分布式节点，所以能展现多个节点的任务信息，包括扫描队列信息，正在扫描数量以及已完成扫描的数量。展示节点状态是running还是pendding。

任务量报告是以柱形图显示的，主要显示每天扫描的任务量，有一个直观的比较。任务量报告的基本单位可以以天、周、月、年等作为比较的单位。

漏洞统计为扫描端扫描到有相关威胁的漏洞信息后，统计其相同类型的威胁信息数量，以不同颜色的区分展示出来。



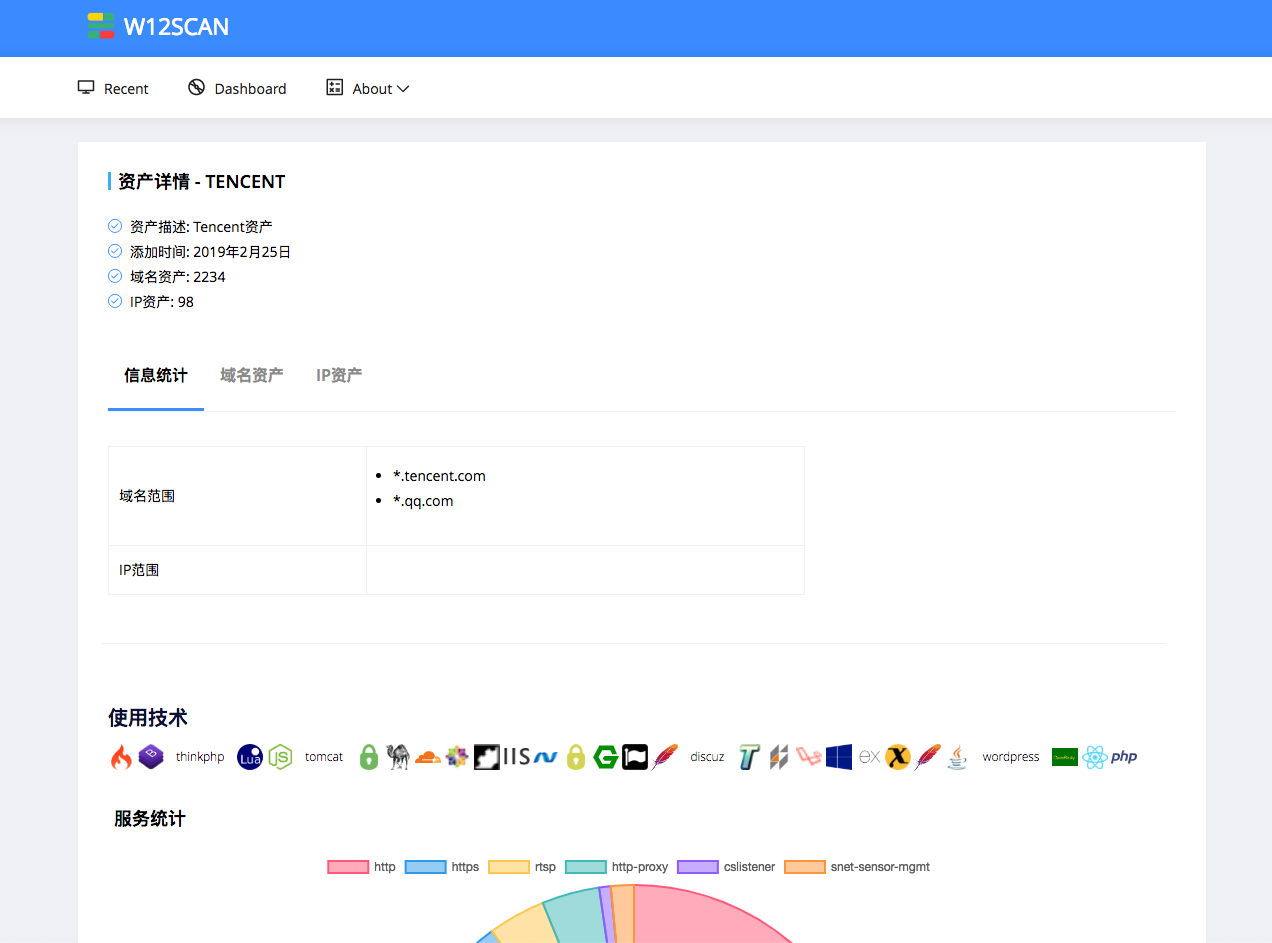
图**4.2**  导航布局

大体设计图如图4.2所示。

4.2.1.3 资产布局

资产布局用于显示IP资产，域名资产或公司相关资产的布局，每个布局结果大体相似，但略有些许不同。资产布局的显示通过tab页面来展示，tab栏目分为三层，基础信息，域名资产信息，IP资产信息。基础信息用于展示该资产的一些基础信息，如资产的名称，资产描述，资产ip范围等等，下面会通过图表的形式来总结整个资产影响的范围，各类应用的占比（通过图表的方式显示）。剩下的域名资产信息和IP资产信息都通过表格来展示。

资产布局，主要显示出该资产下的IP与域名使用了哪些技术，以及有多少存在威胁的网页，资产统计后有“域名资产”以及“IP资产”的统计表格，会将该资产下的信息以表格的形式展现出来，同时你可以点击查看详情来该资产下域名或者IP的更多信息。



图**4.3** 资产布局

如图4.3所示，域名资产的表格内容包括域名，IP，标题，使用技术和附加信息，ip资产表格的表头信息有ip、开放端口和额外信息。

4.2.2 API模块设计

扫描端(Client端)需要与elasticsearch数据库交互，中间通过web的restful api进行沟通。API模块的设计就是解决这层中间问题。

主要两个接口，一个用于接收域名信息，一个用于接收ip信息，使用POST方式提交，内容为json格式。考虑到api的安全性，使用一个验证头（header）来校验。

4.2.3 资产模块设计

资产模块设计包括资产的增删改查入库操作，并且配合elasticsearch搜索引擎展现资产详情。资产的增删改查用MySQL数据库存储，资产模块需要字段有资产名称、资产描述、ip范围、域名范围和添加时间。在设计中，通过ip范围和域名范围，使用elasticsearch过滤出需要的数据。

4.2.4 搜索模块设计

搜索模块的设计中，可以根据相应的搜索语法搜索相关数据，通过类似key="value"的语法搜索，elasticsearch存储的是非关系型数据，因为存储的字段比较多，所以用此搜索方式来搜索数据。搜索模块同时也支持组合查询。相关搜索语法示例：

•title=“abc” # 从标题中搜索

•header=“abc” # 从http头搜索

•body=“123” # 从body搜索

•url = “.baidu.com” # 搜索baidu.com的子域名，``号是通配符

•ip = ‘1.1.1.1’ # 搜索IP，支持CIDR'192.168.1.0/24'和通配符搜索'192.168.1.\*'

•port = ‘80’ # 搜索端口

•app = ’nginx’ # 搜索组件

•country = ‘cn’ # 搜索国家

•service = ‘mysql’ # 搜索服务

•bugs = 'xx' # 寻找存在某个漏洞的地址

4.3 数据存储结构

对搜索服务的需求使用Elasticsearch服务框架，Elasticsearch是一个基于的开源搜索引擎。无论在开源还是专有领域，Lucene可以被认为是迄今为止最先进、性能最好的、功能最全的搜索引擎库。所以在数据字段的定义上需要依据Elasticsearch的规则。Elasticsearch是一种非关系型的数据库，常见的数据类型有intger整数型，text文本型和keyword关键词，date日期型。Elasticsearch默认会对text类型进行分词解析，以保证极速的查询速度。

Elasticsearch和关系型数据库的定义数据存储结构的方法完全不同，所定义的是一些在系统实现中必须的字段，也是需要被“分析”的字段。类别关系型数据库，Elasticsearch有自己Index和Type，Index 就像关系型数据库里的 database, Type 就像 database 里的 table。虽然这并不准确。

在Elasticsearch系统中主要存储两个type

4.3.1 域名信息存储结构

域名信息主要存储状态码，标题，header头，网页源码，服务器文本，IP地址，URL地址，CMS名称，添加时间。因为是Elasticsearch存储数据结构，数据字段是非关系型，我们只能人为的来确定一个“索引”中将包含哪些字段以及每个字段的含义是什么，Elasticsearch也提供了很多数据类型提供选择，并且数据类型与关系型有所差别，主要体现在定义字段的名称上。用一个图来表示域名字段的存储结构以及相对应的结构类型。

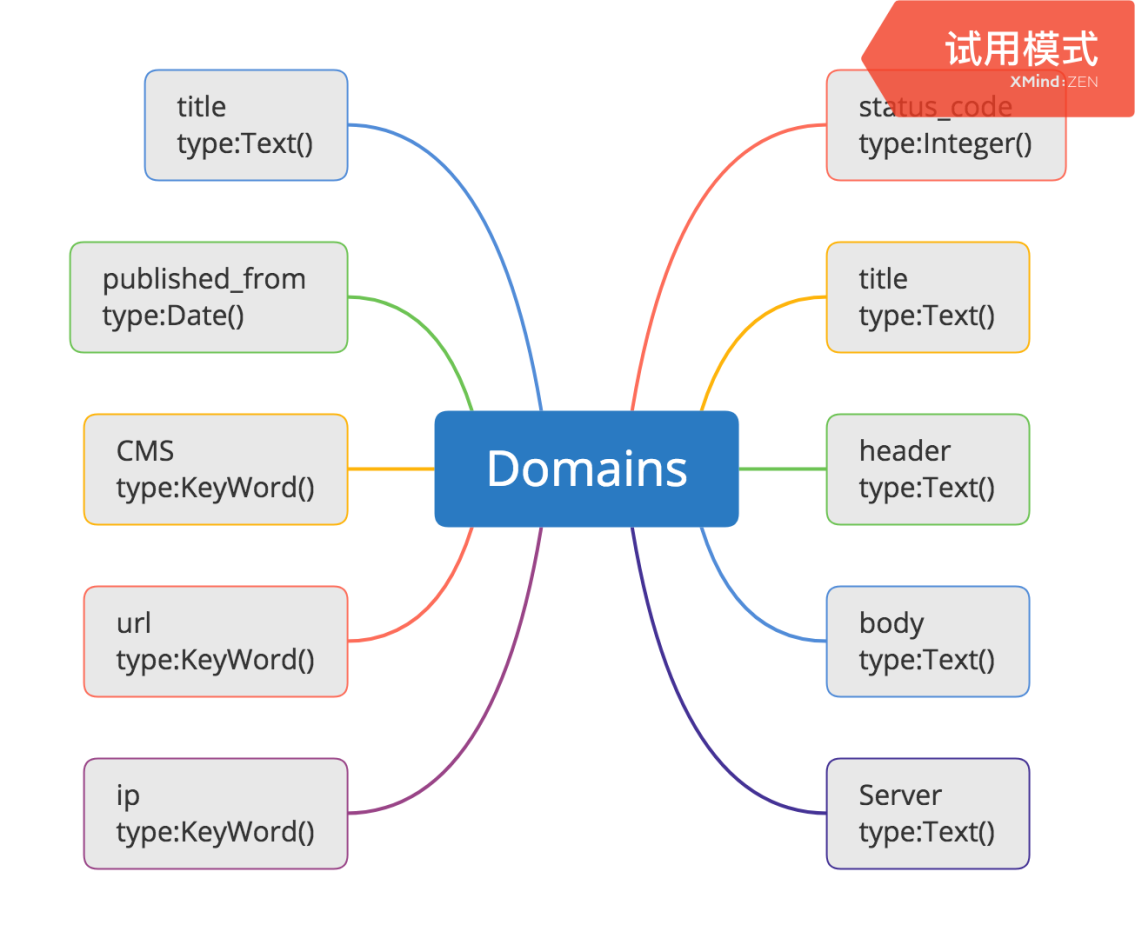


图 **4.4**  域名信息数据结构

域名信息数据结构图如图4.4所示。

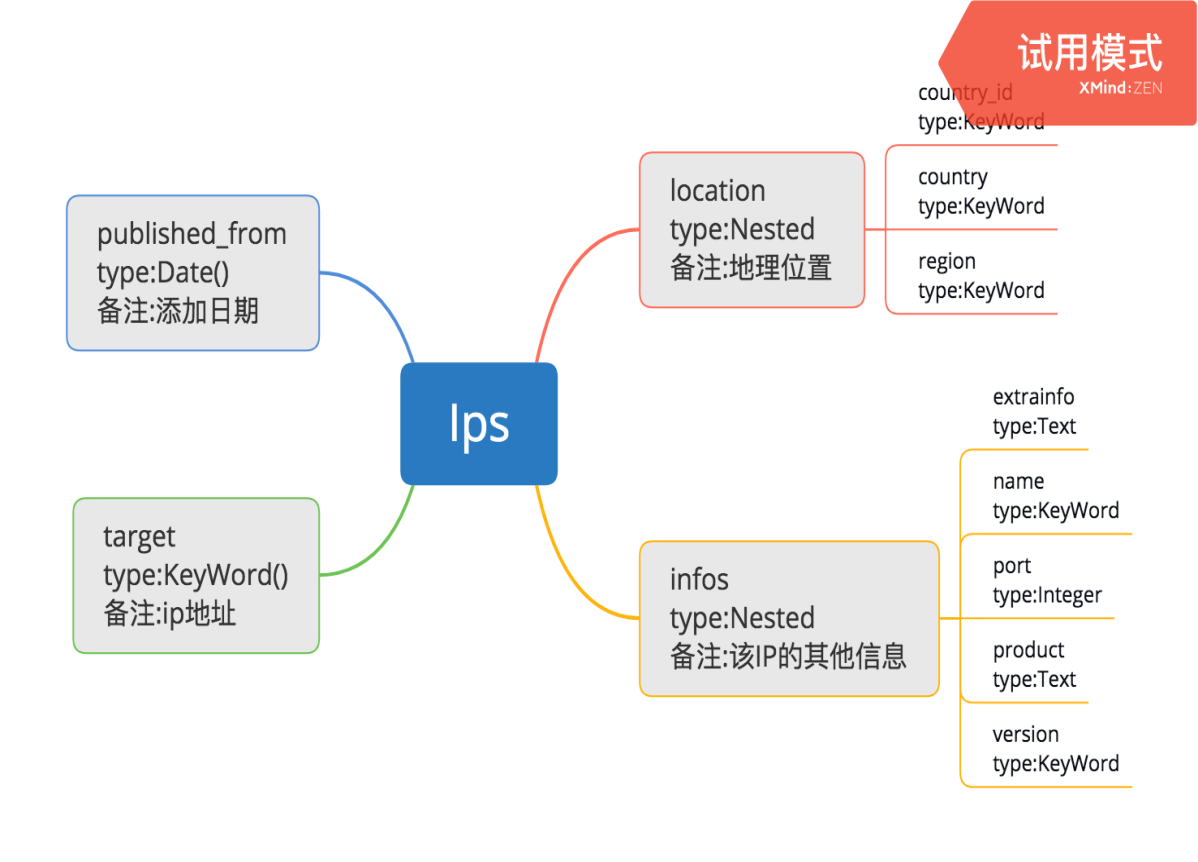
4.3.2 IP信息存储结构

IP信息不仅要存储位置信息和IP的一些基本信息，包括版本信息，服务名称和端口开放信息，因为是非关系型的数据库，有些数据定义字段结合自身业务需求的关联性，需要将数据结构定为嵌套结构，在嵌套结构中在定义多个相关字段。

加入嵌套字段的意义是增强理解度，例如一个地理位置location的字段标识中，还可以下分为国家英文简写码，国家名称，对应城市等等，相应的还有IP信息，对应有该IP下每个端口所对应的信息，这些非关系型的数据用一个“索引”就能很好的存储，如果使用的是关系型的数据库，恐怕就要多建几张表，显得冗余复杂。

Elasticsearch在聚合分析时对嵌入的字段也能够有很多的支持，只需要定义字段的类型为Nested，Elasticsearch在分析时就能进入到嵌入字段内部，对该字段下的下属字段进行分析了。

下图是定义IP信息存储结构中的字段定义详情，包括对嵌入字段的定义，以及相关字段定义的注释等。



图**4.5** IP信息存储结构

如图4.5所示，location与infos定义为Nested嵌套字段，因为在该字段下还有下属的相关联性的字段内容，location有country\_id国家id标识码,country国家名称,region地区，infos字段有extrainfo额外内容，name该IP下对应端口服务名称，port开放端口，product对应IP服务，version版本信息等。

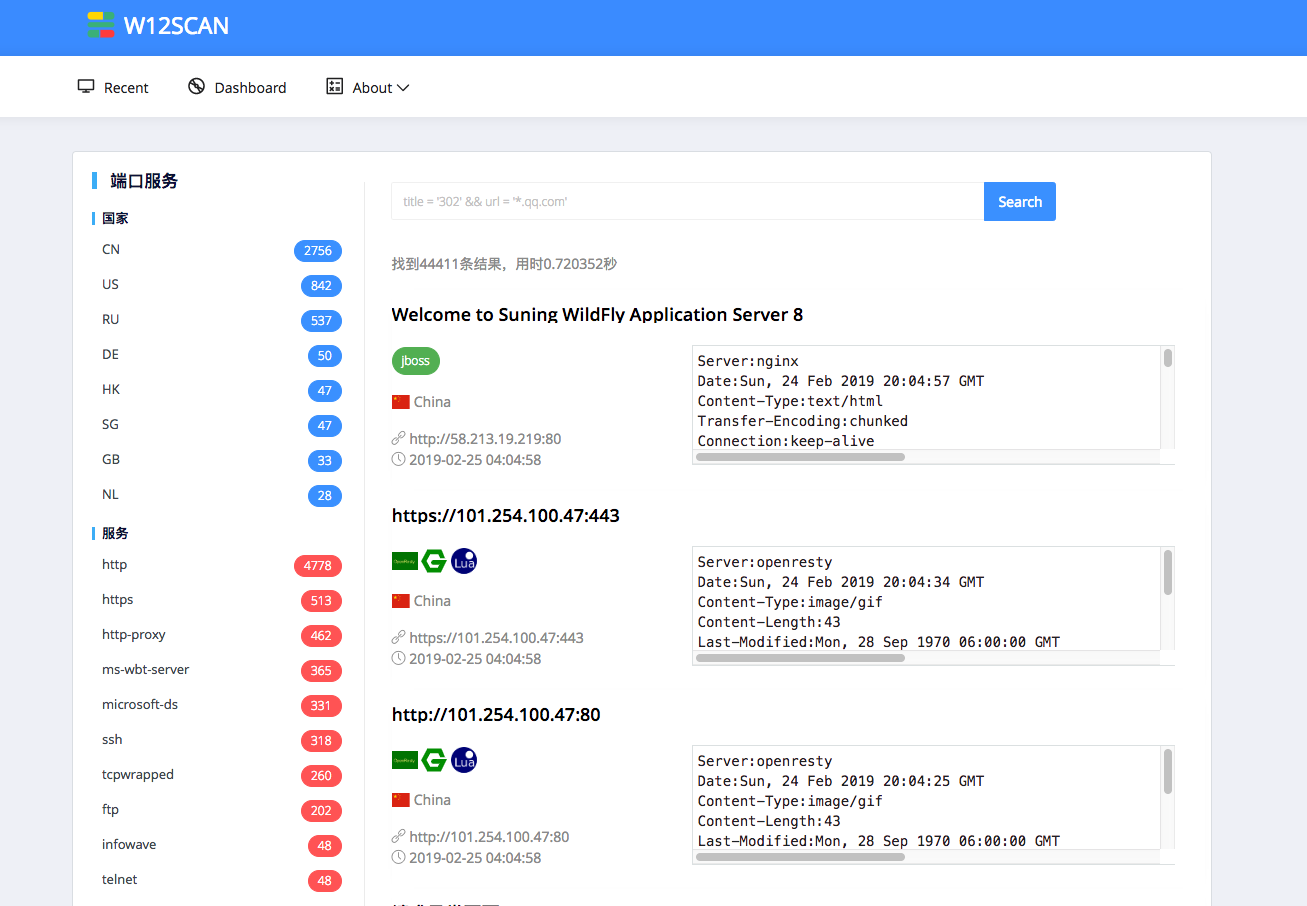
5 运行测试

5.1 WEB端运行测试

Web端运行测试主要展示Web端的功能以及与用户进行的交互操作等，Web端上还有与扫描端进行连接的作用，用户不需要对扫描端进行操作，在Web端上的操作会通过数据库中间件（Redis）来转发给扫描端。

5.1.1 首页

首页会分为三个部分，顶部的导航栏以及下发左侧的统计信息，用于统计国家，端口，服务以及App，清楚标明数量，右侧为展示区，展示新扫描到的地址以及相关的详情。如图5.1 首页所示：



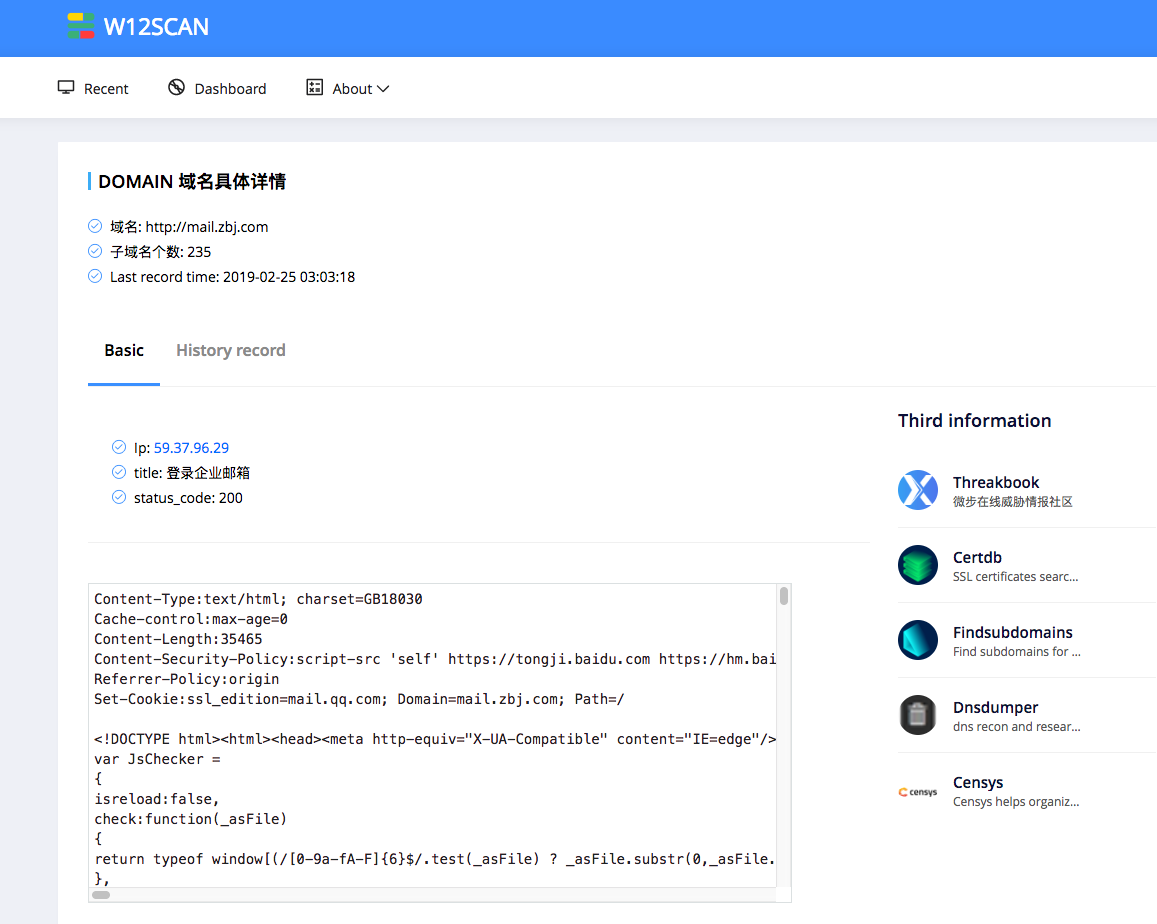
图**5.1**首页

首页的结构排版样式上，参照了shodan，zoomeye，fofa等同类型的网络引擎的样式。但总体来说，和它们还是有很多不同之处，左侧的统计信息，轻轻点击一下就能自动生成相对应的搜索语法展现到搜索框中，在搜索框中点击Search按钮即可搜索到该搜索语法所对应的链接，地址，等详细信息在右侧的界面显示。

右侧界面主要展示了搜索到目标的详细信息，如网址的标题（如果该网址无标题则使用该目标的链接），所使用的应用图标，国家国旗图标，同时还有对应的Http Response返回消息，点击标题处即可进入到该目标的详细域名资产信息中。

5.1.2 域名资产信息

图5.2域名资产信息即展示了该域名的详细资产信息。如果你是通过一个“域名”进入到的该页面，它同时会显示该域名的子域名对应的资产信息，告诉你数量并显示。下图一个选择夹分为Basic基础信息与History record历史记录信息，基础信息中显示ip，标题，使用技术，Http Response返回等信息，同时也提供一些第三方在线查询站点查询相关的信息。

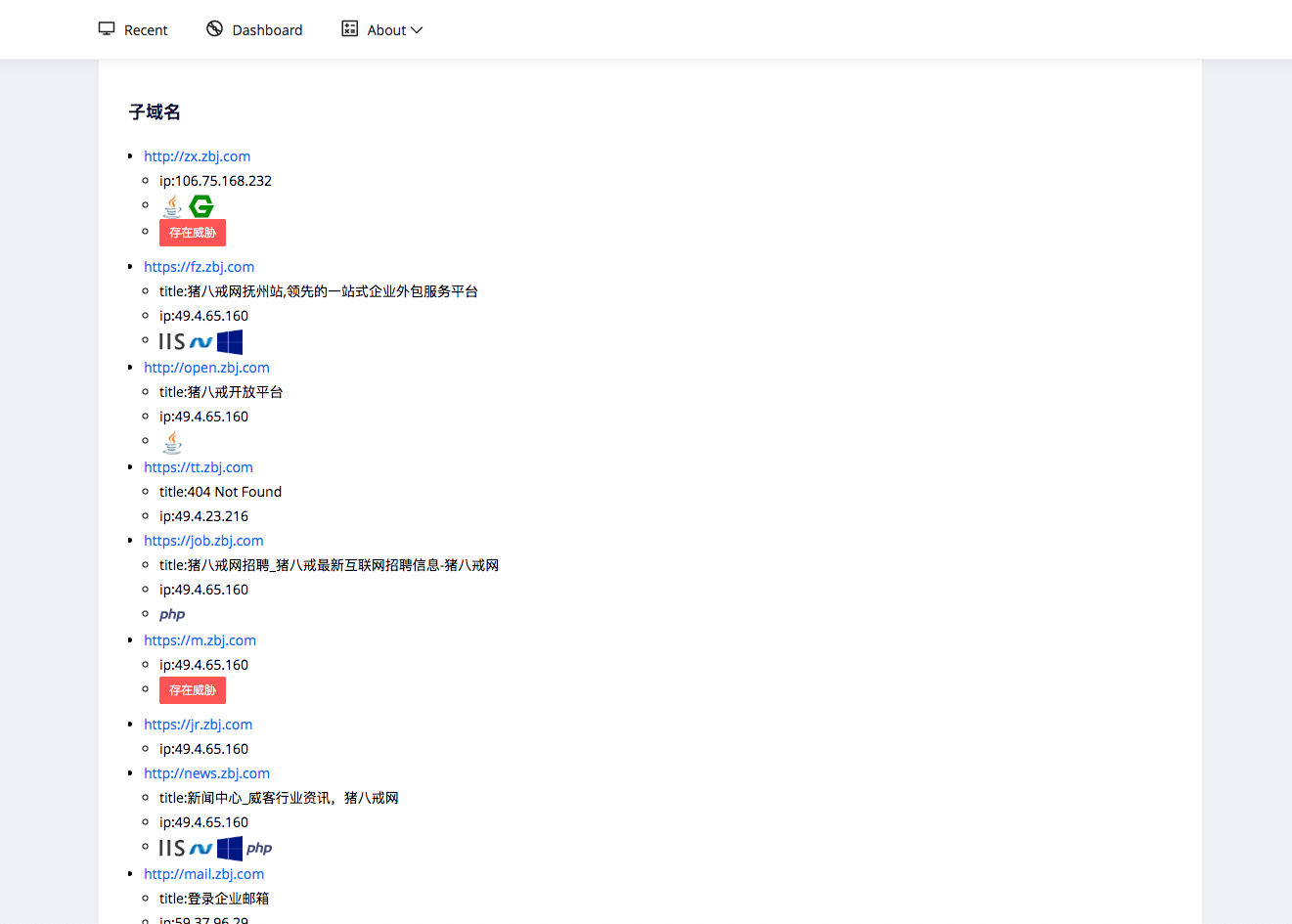


图**5.2** 域名资产信息

History Record历史记录则显示该目标所扫描的历史记录信息，我们知道，一个域名或一个IP，在一段时间内可能是会变得，有时它的历史记录信息也会显得比较珍贵。W12Scan默认会将一个目标缓存一段时间（默认为30天），缓存期限一过，则可以再次添加该目标，同时相关的信息也会被最新的替换，如果想找到历史的记录，到History Record中查看即可。

W12scan也会自动关联子域名上的所有信息，展示标题，IP以及使用技术等相关信息并将可能存在的威胁列举出来。它是w12scan的核心部分，即将相关资产链接，串联起来，是w12scan自动搜索相关资产的核心部分。

如图5.3 自动关联域名所示，从一个域名目标点击进入详情页面后，该目标的所有子域名都会被发现并找出来，该子域名的一些关键特征点也会被展示出来。从中也可以任意点击去到不同目标的详情页面中。



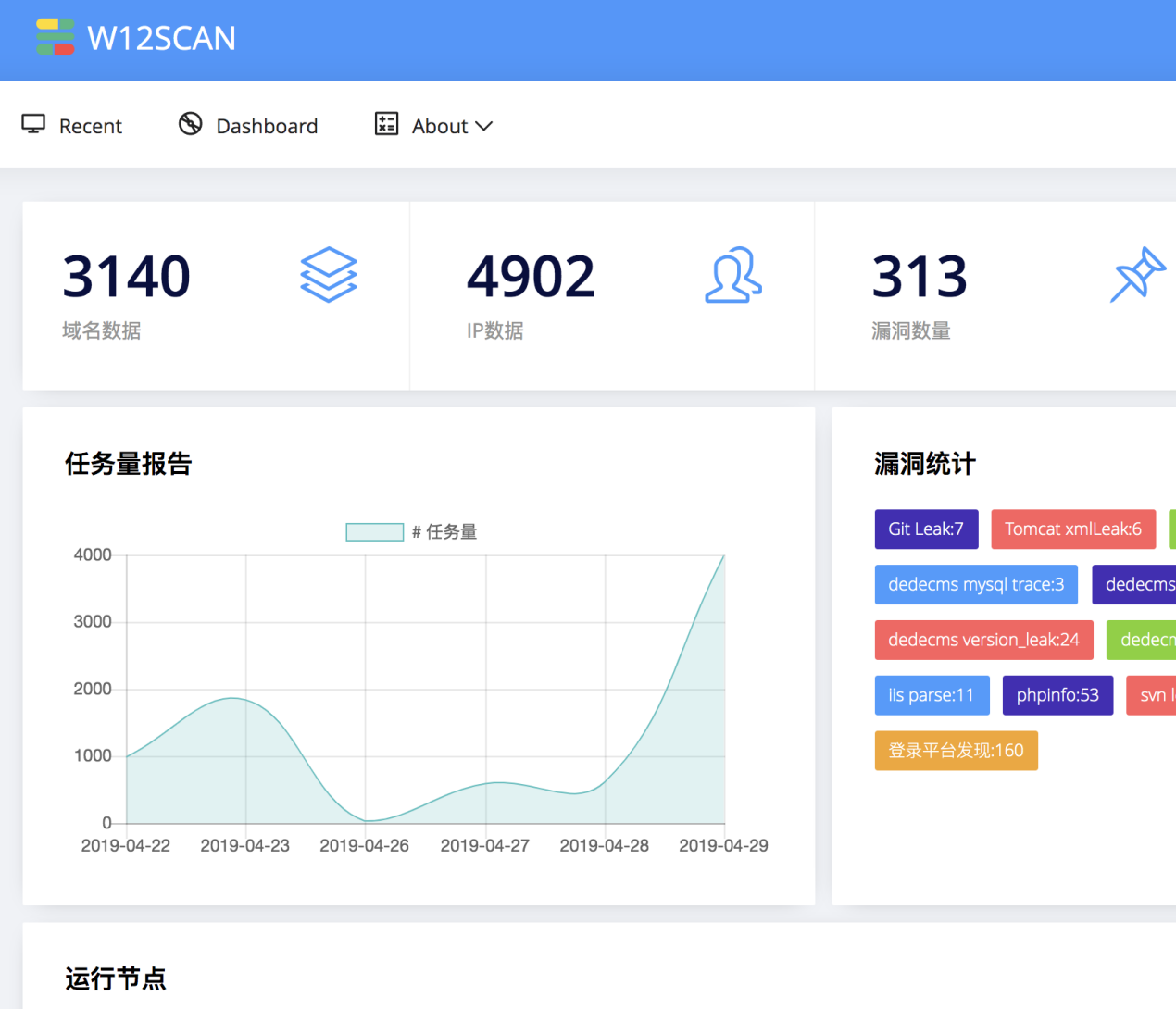
图**5.3**  自动关联域名

5.1.3 系统统计页

系统统计页面相当于整个系统的状态分析，数据分析的起始页面，用于统计扫描了多少域名以及IP，扫描漏洞的情况以及每天扫描的任务量的分析。同时也有运行节点的状态等等。

统计页面的主要功能是提供用户一个参考，让用户了解整个系统的运行情况，方便之后系统的升级以及功能的完善。第一列展示域名数据，IP数据以及漏洞数量信息，第二列展示任务量的报告，以可视化的柱形图展现出来，方便用户了解到所有扫描节点每日的扫描任务总量。

漏洞统计即是使用内置的漏洞扫描框架对每个域名的扫描统计信息，将可能出现的漏洞利用点以名称的形式展现出来，并统计其总量，这只是一个大范围的粗略统计，如果是通过资产配置中进行的漏洞统计，将统计其详细状态与数量，统计会更为细致。如图5.4所示。



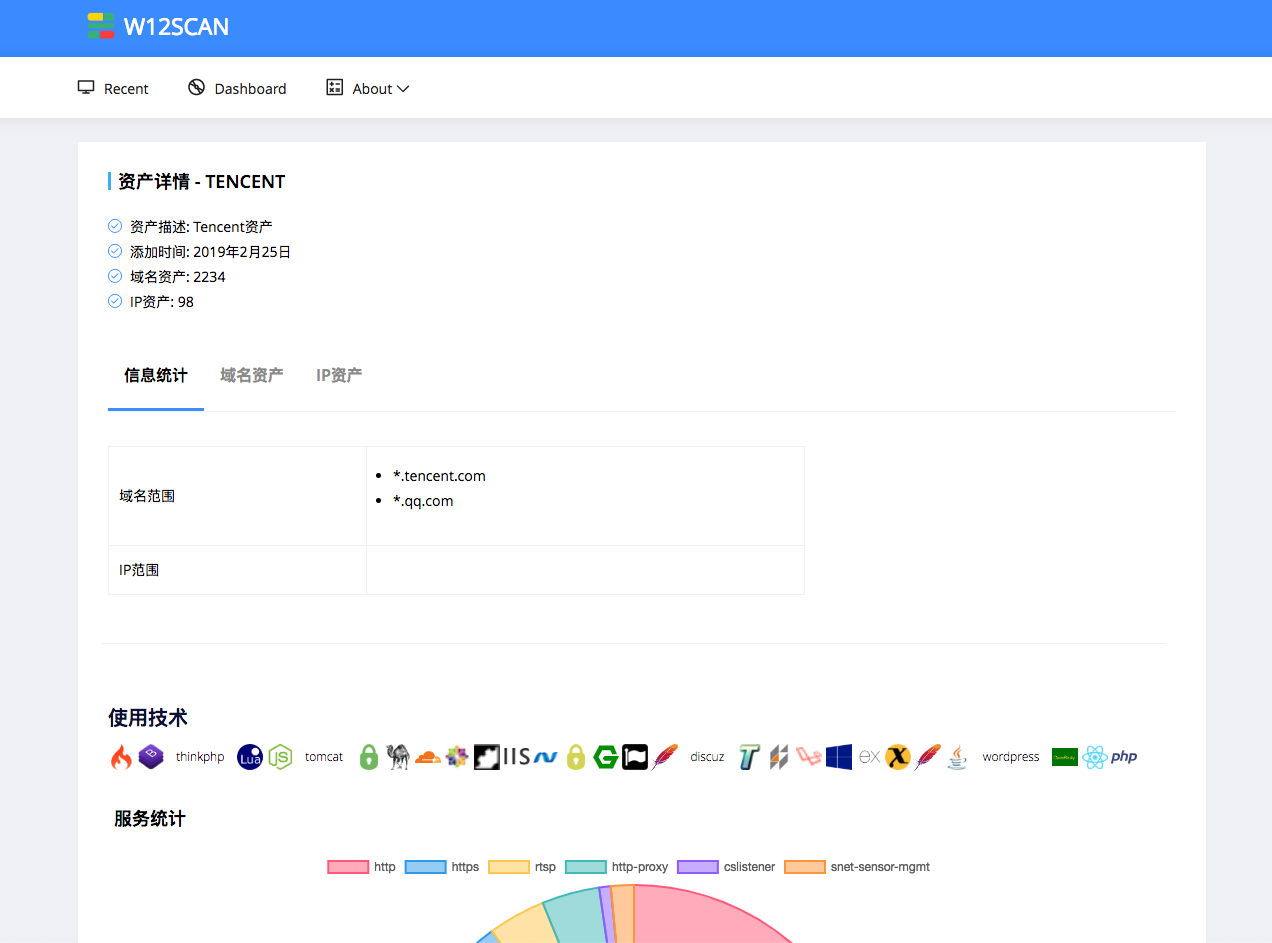
图**5.4** 系统统计

系统统计处也有任务的添加选项以及资产的配置信息等功能，由于篇幅原因，暂无法展示完全。

5.1.4 资产详情

自动查找只是w12scan资产关联中的一部分，你还可以手动将一些关联性的资产加入进来，例如像查找腾讯Tencent的相关资产，只需要在资产配置中配置类似\*.qq.com的域名，w12scan支持通配符进行匹配。当然，Tencent的资产肯定还不止这些，例如\*.tencent.com等等的都可以加入进来，如果又从其他地方找到了Tencent的IP段划分范围，也可以加入进来，资产配置中支持CIDR的IP段描述。加入进来后，w12scan将对相关的域名范围以及IP范围进行聚合分析，得出结果并以图表的形式展示。

资产详情中的信息，也是由tab选择夹的方式展现的，可以由图5.5 资产详情看到，主要分为三层，第一层是信息统计，用于统计基本信息，第二层是域名资产统计，会将聚合分类到的域名资产以表格的形式在此页面中展示，第三层是IP资产统计，也是通过Elasticsearch聚合分类的形式以表格的形式在此页面中统计显示。



图**5.5** 资产详情

资产详情是完全由用户可控配置的，用户也可以配置多个资产详情。在未来的使用场景中，可以想象，你可以配置腾讯的资产，华为的资产，中国湖北省的资产，美国的资产等，w12scan可以分析出该范围内的资产详细情况，漏洞情况，方便安全从业人员对其进行安全监测以及漏洞挖掘等。

5.2 扫描端运行测试

前面说到，扫描端是不直接与用户进行交互的，用户需要从Web端进行交互，Web端将相关数据存入到中间件数据库中，由扫描端从中间件数据库读取消息并按照消息内容执行程序。

整个系统的设计是以docker容器的形式部署，扫描端作为一个后端程序，运行情况不好以截图的形式展现。但是如何查看扫描端的进度以及扫描的过程呢？如何发现扫描节点的异常情况并进行及时的处理？

我们知道，扫描端的作用是为WEB端提供数据的支持。所以在Web端设计了一种可以在web端直接查看扫描节点的运行过程的功能，可查看到当前节点的运行日志状态，如图5.6 节点日志所示。



图**5.6** 节点日志

通过节点日志，可以清楚查看到扫描端的运行过程，运行时间以及运行内容，方便我们即使找到异常点并进行处理。

6 自动化部署

整个项目依赖的组件较多，如果开源给别人使用的话会遇到各种各样的问题，能否将自己的环境打包一份，其他人使用只需要下载这个打包的镜像就好了，而且可以在生产环境使用？Docker可以解决这个问题。

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，而后发布到任何通行的 Linux 机器上，并且能够虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。

6.1 使用docker目的

基于docker编写dockerfile文件，使得本系统可以在任意装有docker的操作系统中运行，而不需要关注其他数据库等服务的安装。

6.2 WEB Dockerfile

WEB镜像基于docker中最小的镜像alpine构建，在此基础之上安装python以及python依赖包，以及django初始化相关的配置。同时安装tzdata同步为Hong Kong时间，总体来说，编译WEB的dockerfile，即相当于在一个纯净的系统上安装我们需要安装的东西，我们将安装步骤记录在一个文明文档中，docker根据该文档会生成一个二进制文件，该二进制文件虚拟了一个虚拟操作系统用于运行我们的应用。

6.3 扫描端dockerfile

扫描端dockerfile基于debian镜像构建，在此基础之上安装python3、nmap、masscan等工具，扫描端也是依赖环境变量进行相关的配置，扫描端只依赖于中间件数据库redis读取消息列表，所以在dockerfile的编写中也要注意扫描端的启动需要在redis的启动之后。

6.4 多个镜像启动

docker的魅力就在于此，我可以每个服务开启一个docker，这些开启的docker各自独立却又彼此关联。WEB、Client、Redis、Elasticsearch各自开启一个docker容器，之间关联的ip通过环境变量读取。docker-compose的语法构建。

多个docker之间的网络是可以互通的，且一些配置的传递如redis和Elasticsearch的用户名密码用环境变量的方式读取并传递。

Docker-compose也可以定义容器的启动顺序，例如在我们这个项目中，web容器需要依赖redis与Elasticsearch容器先启动，client容器需要依赖web容器的启动，在depends\_on中定义即可。

一行命令即可将整个系统启动，并部署，隔离真实环境，提高了效率与安全。

7 总结与展望

资产收集的行为已经成为黑客们信息收集中必备的手段了，随着Zoomeye、Fofa、Shodan的广泛使用，我也曾幻想自己也能写出和它们一样的工具，但现在我将它创造出来了，我把这款网络资产发现引擎命名为W12SCAN，为什么命名为W12SCAN，它不仅是之前W11SCAN的下一代版本，更是我之前写过的W系列扫描器的集合，从W8SCAN到W12SCAN，每款都有各自的特点，每款都有独立的风格。

通过这次设计，因为需求原因接触到了自己之前从未接触的东西，例如Elasticsearch，Django等等，出现过很多问题，幸运的是最后都被解决了。在最后的部署上线过程中，我又学习了Docker的使用，和Nginx与Django uwsgi的联调方式上线网站。当网站成功运行的时候，内心充满了满足感。

7.1 开源

虽然整个系统还像一个孩子一样不够成熟，还有很多功能可以优化，当数据量巨大的时候还会有速度的瓶颈等等，但我相信在未来随着我技术的提高这些问题都会逐步解决，我会将W12SCAN开源到GitHub，期待开源后会有越来越多的安全爱好者加入到我的项目中，不断学习与交流。

开源也不会是这款产品的终点，我也会不断的更新这款工具，每当有个比较好的点子，我也会第一时间将它加入到代码中。并通过GitHub的Issue、Pull Requests、Project Todo等功能不断规划这款产品的未来发展方向，总之主旨只有一个，那就是做出一个开源的，好用的，强大的，让安全研究人员认同的网络空间搜索发现引擎！