清 华 大 学

综 合 论 文 训 练

题目：威胁情报数据融合与推送服务系统

系　　别：计算机科学与技术系

专　　业：计算机科学与技术

姓　　名：黄锐皓

指导教师：尹霞教授

2018 年 6 月 10 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　期：_____

中文摘要

网络漏洞扫描工具是一种用于搜索网络主机中漏洞的工具，它依赖于各种已知漏洞组成的漏洞库。本文利用互联网设备搜索引擎，对其数据进行融合，然后和 CVE 信息进行比对，提出了一种新的检测网络主机漏洞的方法，弥补了扫描工具漏洞库更新不及时等不足，并实现了这样的一个系统，能够监测目标主机上可能存在的威胁，在出现威胁时及时通知用户。我们将在文中讨论本系统的设计与实现中的细节。

关键词：漏洞；CVE；互联网设备搜索引擎；融合；推送

ABSTRACT

Network vulnerability scanners are tools that scan the network to find vulnerabilities on computers. The scanning depends on known vulnerability database. In this paper, we present a new way to find vulnerabilities that uses Internet-connected devices search engines. It aggregates data from different Internet-connected devices search engines and compares them with CVE details. It fixes the disadvantage of scanners that vulnerability database is not always up-to-date. We also implement this system that can monitor computers and notifies users whenever new vulnerabilities are found. In this paper, we will discuss the design and implementation details of the system.

Keywords: Vulnerability; CVE; Internet-connected Devices Search Engine; Aggregation; Notification

目 录

主要符号对照表

HPC	高性能计算 (High Performance Computing)
cluster	集群
Itanium	安腾
SMP	对称多处理
API	应用程序编程接口
PI	聚酰亚胺
MPI	聚酰亚胺模型化合物，N-苯基邻苯酰亚胺
PBI	聚苯并咪唑
MPBI	聚苯并咪唑模型化合物，N-苯基苯并咪唑
PY	聚吡咙
PMDA-BDA	均苯四酸二酐与联苯四胺合成的聚吡咙薄膜
ΔG	活化自由能 (Activation Free Energy)
χ	传输系数 (Transmission Coefficient)
E	能量
m	质量
c	光速
P	概率
T	时间
v	速度
劝学	君子曰：学不可以已。青，取之于蓝，而青于蓝；冰，水为之，而寒于水。木直中绳。輮以为轮，其曲中规。虽有槁暴，不复挺者，輮使之然也。故木受绳则直，金就砺则利，君子博学而日参省乎己，则知明而行无过矣。吾尝终日而思矣，不如须臾之所学也；吾尝跂而望矣，不如登高之博见也。登高而招，臂非加长也，而见者远；顺风而呼，声非加疾也，而闻者彰。假舆马者，非利足也，而致千里；假舟楫者，非能水也，而绝江河，君子生非异也，善假于物也。积土成山，风雨兴焉；积水成渊，蛟龙生焉；积善成德，而神明自得，圣心备焉。故不积跬步，无以至千里；不积小流，无以成江海。骐

骥一跃，不能十步；驽马十驾，功在不舍。锲而舍之，朽木不折；锲而不舍，金石可镂。蚓无爪牙之利，筋骨之强，上食埃土，下饮黄泉，用心一也。蟹六跪而二螯，非蛇鳝之穴无可寄托者，用心躁也。——荀况

第 1 章　引言

1.1　背景

在互联网高度发达的 21 世纪，我们可以享用各种网络服务，比如浏览网页，收发邮件，文件共享等等；互联网产业也在飞速发展，世界市值前十的公司里面有 7 家都是互联网公司，我们每天都在使用它们的产品，给我们的日常生活带来了巨大的变化。但是，“机遇与风险并存”，在我们享用互联网带来的便利的同时，互联网还带来了许多负面的问题，其中一个重中之重就是网络安全问题。

网络安全问题是在互联网中的计算机由于受到恶意的攻击，导致的软件或者硬件受到损害，比如破坏、更改或者泄露所引起的问题。著名的网络安全问题就比如 2017 年 5 月 12 日开始的“永恒之蓝”[1]——全球范围内爆发的基于 Windows 网络共享协议进行攻击传播的蠕虫恶意代码的网络攻击事件，全世界有诸多高校内网、大型企业内网和政府机构受到攻击，造成的损失不计其数。实际上，在该漏洞爆发之前，微软就已经发布了这个漏洞的补丁，但是，用户并没有对自己的计算机上的漏洞有所了解以及重视，才会不幸受到黑客的攻击，如果能提前知道计算机上的漏洞就能未雨绸缪。

而为了解决这样的网络安全的问题，漏洞扫描工具应运而生。漏洞扫描工具是一种检测目标系统存在的漏洞和弱口令的工具，可以监测目标中的通用漏洞。它们一般通过一系列逻辑判断，指出目标上的比如缓冲区溢出等漏洞。著名的扫描工具有 Metasploit[2]，SQLMap[3]，Nessus[4] 等。漏洞扫描工具不仅能扫描外网，因为是可以自己部署的，也可以扫描内网。

但是，既然是扫描漏洞，就必然需要一个漏洞库，现有的扫描工具都存在一个问题就是漏洞库更新不够快，而且扫描出的漏洞也不是有已知编号的漏洞，常见的检测的漏洞比如 SQL 注入，XSS，CSRF 等。同时，漏洞扫描通常是进行渗透测试，可能会对系统产生不可预知的影响；因此进行扫描时，只能对已获得授权的目标进行扫描，不能对“别人”的主机进行扫描。

但是，实际上现在又有一种互联网设备搜索引擎，它们是用一些网络扫描工具，比如 NMap[5]，扫描全网主机上的一些 Fingerprint 信息，然后提供给用户的搜索引擎。而这种搜索引擎每一款用的方法都不一样，产生的结果也就不一样。因此，如果我们能对不同的互联网设备搜索引擎提供的数据进行融合，而不

需要实际去扫描，然后再和已知编号的漏洞进行对比，就能最大限度地发现目标主机上的潜在漏洞。在发现了这些潜在漏洞之后再由用户去实际判断是否真的受到这种威胁，可以大大减轻安全维护的工作，对网络安全的研究有重要意义。

1.2 项目需求

该项目的需求是寻找并实现一种发现主机威胁情报的方法，可以弥补现有扫描工具的一些不足，如只能针对特定漏洞进行扫描，漏洞库不全，更新不及时，不够自动，需要授权，对目标可能产生不良影响等；并且该系统也提供推送功能，在发现漏洞时推送威胁情报信息；也提供可视化工具，用户可以比较容易地操作该系统完成对主机系列的安全监测。

1.3 本文贡献

本文的贡献是基本满足了上述项目需求，设计并实现了一个威胁情报数据融合与推送服务系统。这个系统的主要功能是监测一系列主机（域名、IP 地址段等）的安全情况，发现可能的威胁时及时地告知事先设置的邮箱或者其他联系方式。

该系统的主要特点是：

- 威胁情报库完整并且更新非常及时，时刻和最新最全的 CVE 信息保持一致；
- 系统并不主动地对主机进行扫描和逻辑的判断，而是采用可以找到的互联网设备搜索引擎 API 并对其中的信息进行分析融合；
- 系统会尽可能多地获得主机系列中的全部主机，而对于每一台被发现的主机，系统会穷尽其相关信息并进行信息的融合；
- 主动发现威胁情报并通知，不需要操作者手动部署扫描；
- 系统提供了一个前端网站供直观地观察可能的威胁情报；
- 发现可能的威胁情报，但是并不实际检验是否真的受到该漏洞的威胁。

1.4 本文结构

本文共分为 7 章。

在第一章引言中介绍本文需求提出的背景和本文的工作；第二章相关工作主要介绍目前现有的有关漏洞扫描的工作以及和本文的工作相关的知识背景，比如现有的扫描工具具体有哪些以及它们是如何工作的，互联网设备搜索引擎是什么等，这些背景和后续本文的工作有很大的关系，可以说后续的工作都是以这些相关工作为背景的；第三章系统设计着重介绍系统的整体结构以及每一部分从需求到分析到最终的设计，不注重细节以及实际生产环境中的困难，而是高屋建瓴地介绍系统的结构与设计；第四章是第三章的后续，从代码的角度介绍系统在实现过程中采用的方法与在实际生产环境中遇到的困难是如何克服的，以及如何提高效率和正确性；第五章是结果检验，将介绍系统实现完成之后的特点、优势、使用方法，以及举例说明系统的正确性与用处；第六章的总结与展望将对本文的工作进行总结，得出结论并提出本文工作的不足与可以进行改进的部分；在最后一章致谢中将感谢为本文的完成提供帮助的人们。

第 2 章　相关工作

2.1　漏洞扫描工具

漏洞扫描是利用已知的漏洞数据库，对本地主机或远端主机进行脆弱性检验的一种方法，是一种发现可利用漏洞的渗透测试，经常和防火墙和入侵检测系统结合使用以提高网络的安全性。管理员可以根据漏洞扫描的结果来发现系统中的漏洞或者错误，在黑客攻击前做好防范的工作，避免不必要的损失。和被动防御的防火墙和入侵检测系统相反，漏洞扫描是一种主动防御方式，可以提前发现问题，做到未雨绸缪。

而漏洞扫描工具就是可以进行漏洞扫描的工具，主要分为针对网络的、针对主机的和针对数据库的漏洞扫描工具，而本文主要关注的是针对网络的漏洞扫描工具，意即通过网络来扫描远程计算机上的漏洞的工具。

而这之中比如 Metasploit 就是一款著名的网络漏洞扫描工具，它提供了数百个已知漏洞的利用工具，可以发现目标主机上是否有相应的漏洞。在发现新漏洞时，其用户可以将漏洞添加到 Metasploit 的目录上，任何使用该工具的人都可以用来检测特定系统是否受到该漏洞的威胁。Metasploit 使用门槛低，易推广，为漏洞检测的自动化和及时化做出了不可小觑的贡献。

其他地，也有 SQLMap 或者 Netsparker 这样专门针对 SQL 注入等特定的类型的漏洞进行扫描的工具，还有像 Nessus 这样的全面的系统漏洞扫描工具和分析软件。这些扫描工具都以漏洞库为基础，通过漏洞利用工具或者一系列逻辑判断，最终找到目标主机上的漏洞的工具。

2.2　互联网设备搜索引擎

2.2.1　Censys[6]

Censys 是一款互联网搜索引擎，它使用 ZMap[7] 每天扫描整个 IPv4 地址空间，传言只需要 45 分钟就能对整个地址空间进行一次扫描，然后搜集所有的互联网中的设备的信息，返回一份关于各种资源配置和部署信息的报告，而所谓资源就分为设备、网站或者证书。

Censys 的官网上是这么介绍自己的：“Censys 是一款给互联网安全从业人员

提供的平台，它可以帮助发现、监视、和分析互联网中的设备。我们有规律地探测所有的公共 IP 地址、热门域名，选择、管理和丰富这些搜索结果，然后通过一个搜索引擎和一套 API 让它变得易于理解。”

2.2.2 Shodan[8]

Shodan 经常被称为最可怕的搜索引擎，或者比 Google 更恐怖的搜索引擎。它也是一款互联网设备搜索引擎，它可以搜索互联网中的各种设备，不仅仅是个人电脑，甚至包括摄像头、打印机、路由器、红绿灯、粒子回旋加速器等等，只要是连接到互联网中的设备，都有可能被 Shodan 发现。如果这些设备没有做好安全防御措施，就都有可能被 Shodan 随意进入。比如说默认密码，如果在 Shodan 上搜索“Default Password”，会发现有不计其数的打印机的仍在使用默认用户名和密码，还有很多其他设备进入根本不需要认证，只需要用浏览器访问即可进行连接。

在 Shodan 的网页上，用户可以搜索一个域名、IP 地址段、甚至一种设备的名称，例如 Webcam，然后找到和关键词相关的互联网设备的信息，比如运行的服务、地理位置等。通过这些信息，就可以分析找到互联网中的脆弱部分，侵犯用户的隐私与利益，或者作为管理员防患于未然，保护用户的利益。

2.2.3 ZoomEye

类似于 Censys 和 Shodan，ZoomEye 也是一款互联网设备搜索引擎，而且是一款国产的互联网设备搜索引擎，中文名是钟馗之眼。ZoomEye 通过搜索引擎开放了他们的海量数据库，用户可以通过搜索得到主机的网站组件指纹（如操作系统、Web 应用、Web 服务等）和主机设备指纹。不过，为了不被黑客大规模利用，ZoomEye 还没有完全开放，目前返回结果的条数受到了一定限制。

2.3 CVE

CVE 的全称是 Common Vulnerabilities & Exposures，中文名是公共漏洞和暴露。CVE 类似于一个字典表，给目前已知的漏洞一个统一的名字，形如 CVE-XXXX-XXXX，中间 4 位是年份，最后的几位（不一定是 4 位）表示实际的编号。通过这个统一的名字，安全工作人员可以在各自的漏洞库和评估工具中共享数据，易于管理。即便如此，不代表这些不同的工具就易于整合到一起，CVE 只是提供了一个目录，或者“关键字”。

CVE 的官方网站是 https://cve.mitre.org，上面提供了每一个 CVE 的相关信息、统计信息，用户也可以自行上传 CVE 或者下载 CVE。而 CVE Details 是一个经过整理过的网站，上面记载了所有 CVE 的详细信息，便于用户阅读，网址是 https://www.cvedetails.com。

2.4 Exploit Database[9]

Exploit Database 是一个漏洞库网站，上面有海量的漏洞利用脚本，安全人员可以利用它来提高公司设备的安全性，网址是 https://www.exploit-db.com。其 Exploits 版块有 Remote Exploits，用于展示最新的远程漏洞利用脚本；Web Application Exploits，用来展示最新的 Web 应用漏洞利用脚本，比如 SQL 注入等；Local & Privilege Escalation Exploits 等。除此之外，还有 Shellcode，Papers 等版块。

第 3 章　系统设计

3.1　总体结构

3.1.1　系统设计目标

为了满足第 1.2节中描述的需求，该系统的主要设计目标是：

- 可以从互联网设备搜索引擎 API 中获得主机系列（域名、IP 地址、IP 地址段等）的信息，如 IP 地址、操作系统信息（操作系统名称、发行版及版本号）、地区信息（国家、地区、城市等）、服务器信息（服务器框架名称及版本号）、软件信息（软件名称及版本号）等；
- 可以对主机信息进行信息融合，对相同主机的信息进行合并，对不同主机的信息进行再次搜索后合并；对同一台主机的不同数据源的信息进行合并时，不同字段的数据进行合并；
- 可以从 CVE 相关网站上获得 CVE 名称列表，并用爬虫对 CVE 的信息进行数据爬取，获得 CVE 的名称、描述、CVSS[10]（通用漏洞评分系统）分数、产生威胁的端口、产生威胁的软件信息（软件名称、版本号、制造商、软件类型等）、相关网页等；
- 可以对主机信息和 CVE 信息进行对比分析，如端口信息、软件信息等，找出可能产生的威胁；
- 可以把从 API 和网站上获取的主机信息、CVE 信息和可能的威胁情报信息存储到数据库中，并在需要的时候进行读取；
- 可以由用户设置监测的主机的范围（域名、IP 地址、IP 地址段等），定时更新主机信息、CVE 信息，并重新分析可能的威胁情报，如一天、一星期甚至一个月；
- 每次发现新的威胁情报之后，通知事先设置好的邮箱或者其他联系方式，该次分析发现了哪些威胁信息；
- 提供一个前端，可以直观地看到设置的主机系列的主机上有哪些主机存在威胁，并且尽可能提供威胁的原因，也能直观地观察该主机的全部信息，如采用列表等；
- 在可能的威胁情报过多时，前端提供过滤功能，可以搜索某一特定 CVE，系统返回可能受到该 CVE 威胁的主机列表；也可以搜索某一特定 IP 地址

（用户想监测的主机范围内的），系统返回这个 IP 是否受到某些 CVE 的威胁，如果不存在威胁的话就不会看到结果；也可以搜索某一主机的范围（域名、IP 地址段等），系统返回有关这个范围的主机的威胁情报信息；

- 当在发现主机上存在威胁时，从一些特定网站上获得该漏洞的检测脚本，在前端上显示。

3.1.2 系统工作流程

图 3.1是系统的工作流程图。

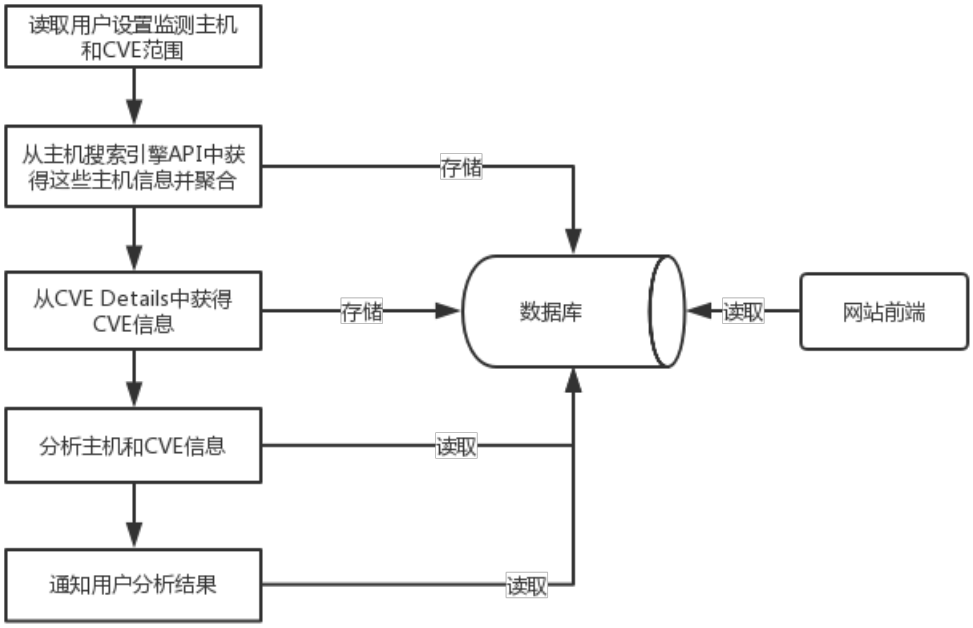

图 3.1　系统总体工作流程图

可以看出，在每次更新时，首先读取用户设置的监测主机和 CVE 的范围，其中主机范围可以是 IP 地址段（如：166.111.0.0/16）、域名（如：tsinghua.edu.cn），或者 IP 地址（如：166.111.176.55）；CVE 的范围可以使全部、某些年的，或者某些指定的 CVE。

接着，系统从互联网设备搜索引擎 API 中获得与这些主机范围相关的主机的信息，这些信息会返回许多台主机的信息，然后由系统应用合并的算法对这些信息进行合并融合，最后返回一个主机信息的并集。这些主机信息的并集存

储在数据库中以备之后的使用，这些信息将会在分析威胁情报和显示前端的时候进行读取。

下一步是进行 CVE 信息的获取，在第一步中已经获得了需要监测的 CVE 的范围，因此这里只对这些 CVE 进行数据的爬取。这里采用的是爬虫的方法，对网站 CVE Details 进行网页页面的解析，获得其中的有用信息，并将这些信息存储在数据库中，这些信息将会在分析威胁情报和显示前端的时候进行读取。

接下来由系统读取之前的主机信息和 CVE 信息并对信息进行比对，通过主机的信息，如开放端口、安装软件等 Fingerprint 来判断任意一台主机是否有可能受到某些 CVE 的威胁。如果有，则将这台主机的可能受到威胁的信息存储到数据库中，如 CVE 名称、CVE 描述、相关信息（References）、可能受到威胁的原因等。这些信息将会在推送和显示前端时进行读取。

分析完威胁情报，系统将会对这些威胁情报进行推送。用户可以在一开始设置主机范围和 CVE 范围的文件中设置希望收到推送的邮箱，也可以同时设置重点关注的 CVE 名称。在这一步中，系统将会把本次分析的全部结果，和每个用户关注的 CVE 的分析结果发送给这些邮箱。

最后还有一个环节是网页前端，这个前端是对存储了主机信息、CVE 信息和威胁情报信息的数据库的一个直观的表示。在前端中，用户可以看到每一台主机的可能受到威胁的 CVE 的信息，如果需要，也可以看到原始的主机信息，也可以进行更高级的搜索，如：搜索 IP 地址、IP 地址段、域名、CVE 名称等。

3.2 网络主机信息获取设计

3.2.1 互联网设备搜索引擎 API 介绍

在相关工作中已经介绍过互联网设备搜索引擎的功能，它们会扫描整个互联网，而用户可以通过一些关键词来找到自己想要的互联网设备的有关资源配置和部署信息的报告。而所谓这些搜索引擎的 API 就是这些网站的开发者开放给其他开发者的一套提供和网站基本相同功能的 API，开发者可以通过这些 API 开发自己的软件。

而本系统最重要的一个步骤就是对主机信息进行融合，而这些信息的来源不是来自主动的扫描，而是来自这些搜索引擎的 API，这些 API 的信息又有所不同（这也是本文工作意义的主要原因之一），因此对这些 API 进行各自提供的信息的对比将是进行主机信息获取前的第一步。本文进行信息融合选取的 API 是

Censys、Shodan 和 ZoomEye，因此在下文中主要对这三种 API 进行对比分析。

3.2.2 Censys API

Censys 提供了一套 REST API 和 Python 中相应的模块，可以通过搜索关键词进行主机信息的获取，如 IP 地址、IP 地址段、域名等，还可以自定义一些字段的值，如在搜索“80.http.get.status_code: 200”时将会搜索 HTTP 协议开放 80 端口的 GET 的返回值为 200 的主机，同时，Censys 还提供了布尔表达式、范围（Range）搜索、正则表达式搜索等诸多功能。Censys 的 API 文档如下：https://censys.io/api。

不过，对于一个 API，最关注的内容还是其提供的信息内容。我们主要关注的 Censys 的 API 有两个，一个是/search，一个是/view。/search 可以通过关键词搜索信息，Censys 会返回一个列表给用户，每一个结果的字段由用户指定，关键词的语法如上文所述，支持主机 IP 地址（/search/ipv4）、网站（/search/websites）和证书（/search/certificates），这里主要关注的是主机；/view 是在用户已经明确知道主机的 IP 地址或者域名之后，可以查看该主机的完整信息，同样支持 IP 地址（/view/ipv4）、网站（/view/websites）和证书（/view/certificates），这里主要关注的是主机。在使用/search 搜索的时候，需要哪些字段需要用户指定，如果不指定 Censys 只会给每个结果返回一些摘要信息，而这些信息是不够用的。因此，往往需要结合上述两个 API 来使用。

不论是搜索还是详情的 API，Censys 返回给用户的主要是运行在这台主机上的开放端口上的服务的 Fingerprint 信息，如运行在 80 端口上 HTTP 服务的 GET 请求的返回值、状态值、文件头等；还有如运行在 22 端口上的 SSH 服务的密钥交换的参数、加密算法等。不过，对于需要自动化的融合工作来说，这就需要对每一种协议做分别处理，因为对于每一种协议，Censys 返回的数据的结构是不同的，如果不能有效地提取需要的信息，就是没有意义的。为了更加有效地使用 Censys，本文仅用/search 来获得更多的 IP 地址，因为相比另外两个 API，Censys 返回的结果条数往往都更多，但是对于其返回的 Fingerprint 信息，则仅取其端口信息和运行的操作系统信息。

此外，在校园网环境下 Censys 的访问速度较慢，往往需要 10 秒钟以上的时间来返回一次查询，而且如果需要大量的查询次数的话需要付费。

3.2.3 Shodan API

Shodan 同样提供了一套 REST API 和 Python 中的模块，也相应地提供了自己的语法，比如搜索“net:166.111.0.0/16”搜索网段信息等。Shodan 的 API 文档如

下：https://developer.shodan.io/api。

而对于 Shodan 返回的信息，它的格式要更加地统一。我们关注的 API 主要有两个，一个是/host，类似于 Censys 的/view，可以查看一台主机的全部详细信息；一个是/search，和 Censys 的/search 类似，也支持自己的语法。在/host 返回的结果中，Shodan 会返回一些基本信息，这些和 Censys 是类似的，如地理位置、ASN、组织、ISP 等等，更重要的是它提供的各种服务的 Fingerprint 信息更加格式化，在“data”字段中，Shodan 提供了运行在其上能找到的所有服务的信息，而把协议名称、传输层协议、运行的软件名称及版本、对应域名等都用相同的字段的名字存储，在自动化处理的时候十分方便。类似地，/search 的结果也只给每个条目一部分基本信息，比如“data”字段只返回了第一个服务的结果，因此同样需要结合/host 使用。此外值得一提的是，在每一种服务里面，Shodan 还提供了本机可能受到威胁的 CVE 的列表，这和本文想做的工作本质上是一样的，只是本文将扩充这个列表，从其他数据源获取数据，然后与这个列表求并集。

此外，速度方面，Shodan 访问比 Censys 快。返回结果数方面，结果条数比 Censys 少不少。权限方面，可以免费申请教育账号，获得更高访问权限。

3.2.4 ZoomEye API

ZoomEye 同样提供了一套 REST API 和 Python 中的模块，但是提供的数据要更少，不过其中也有另外二者没有的结果条目。不过，和前两者不一样的是，ZoomEye 没有提供类似于/view 或者/host 类似的 API，只能一次搜索一系列的主机，并在结果里直接返回主机的相关信息。ZoomEye 的 API 文档如下：https://www.zoomeye.org/api。

我们主要关注的 ZoomEye 的 API 有/host/search 和/web/search，它们都是通过搜索 ZoomEye 自己的一套语法之后返回一系列主机，然后这两个 API 分别返回主机上的基本信息（地理位置、开放端口、操作系统等）和 Web 应用信息（应用、容器、框架、防火墙等）。因为本文的对比方法主要是通过主机上的应用和可能受到 CVE 威胁的软件进行对比得到结果，所以实际使用中使用了/web/search 这个 API。

此外，ZoomEye 的访问速度也较快，但是返回的结果只返回其中的 30%，条数比较少。

3.2.5 各 API 对比表格

综合以上三小节的内容，表 3.1是这三个 API 的对比分析表格。

表 3.1　Censys，Shodan，ZoomEye API 对比

对比项	Censys	Shodan	ZoomEye
访问速度	慢	快	快
结果数目	多	一般	少
支持语法	支持	支持	支持
选用 API	/search/ipv4	/search, /host	/web/search
主要内容	指纹信息	指纹信息	安装软件信息
格式化	较复杂	较规则	较规则
特殊信息	详细指纹信息	可能的威胁	软件信息分类
权限	250 次/月	学生账号不限	只返回 30%

3.2.6　主机信息获取策略

结合以上分析，本文获得主机信息的方法是从互联网设备搜索引擎的 API 中搜索，对于每种目标（一种目标为一个关键词，关键词为 IP 地址、IP 地址段或者域名），对于不同类型的目标，对每种 API 分别采用不同的搜索语法，对结果应用融合，然后把结果存储到数据库。

具体地，为了获得尽可能多的主机上的软件信息、服务信息和基本信息，本文对于三种 API，主要使用 Censys 获得更多的主机地址，仅使用/search/ipv4 来搜索满足关键词条件的主机列表；用 Shodan 来提供详细的指纹信息，首先使用/search 来获取满足条件的主机列表，然后用/host 获取每一台主机上的详细指纹信息；用 ZoomEye 来提供更加丰富的软件信息和版本，仅使用/web/search 来获取满足关键词条件的主机列表中主机上的软件信息。

3.3　CVE 信息抓取的设计

为了分析每一台主机可能受到的威胁，把主机和有固定编号的 CVE 进行联系，就需要每一个 CVE 的相关信息。和第 2.3节中提到的一样，CVE Details 就是这样一个记录了每一个 CVE 的详细信息的网站，也处于实时更新中。对于每一个 CVE 单独的界面，它提供了该 CVE 的名称、简介、CVSS 分数、威胁类型、可能受到威胁的软件信息、一些相关阅读等。

因为本文采用的判断是否有受到威胁的可能的方法是对比主机上安装的软件和可能受到 CVE 威胁的软件，和开放端口和可能受到威胁的端口，所以在抓取 CVE 信息时，安装软件的信息和可能的端口信息就是必不可少的。然而，因

为比对软件名称和版本号的时候无法进行精确的比对，因为软件名称可能有多个叫法，软件版本也存在细小的差别，比如 Beta 版本等，仅仅进行模式匹配往往是不够的，因此，在发现可能的威胁的时候，具体的判断最终往往还是由人来进行，这就需要把 CVE 的描述也放进爬取的内容之中。当用户发现这台主机有风险的时候，可以一个一个阅读 CVE 的描述，把匹配出错的 CVE 排除，只留下真正的威胁情报。类似地，有的 CVE 虽然是真正的威胁，但是 CVSS 的评分非常低，可能只有四五分，这样的漏洞即便最强大的黑客也难以利用，但是它又是真的存在，因此，CVSS 分数也是爬取的一部分，用户看到该 CVE 的 CVSS 分数很低，即便判断正确了，也可以不管。

此外，在具体抓取时，注意到 CVE Details 并没有提供 API，而是只呈现了一个网页，所以本文采用爬虫的方法对 CVE Details 进行爬取。

而实际在爬取时，可能产生威胁的软件由网站提供了一个列表，是非常格式化的，可以通过爬虫获得详细的内容，但是端口却没有这样一个列表，而是隐藏在 CVE 简介中。因此在进行端口爬取的时候，如果采用自然语言处理的方法比较困难，实际采用的方法是简单的匹配，用正则表达式匹配几种发现的端口出现的方法，比如用“port (\d*)”匹配“port 443”，又如用“(\d*)/tcp”匹配“443/tcp”，用这种正则表达式匹配的方法可以找到绝大多数的端口信息。

综合以上分析，CVE 信息抓取的方法是用爬虫对 CVE Details 的网站进行爬取，爬取的内容包括每一个 CVE 的名称（编号）、可能有威胁的端口、可能有威胁的软件（名称和版本）、CVSS 评分和简介。在爬取了目标的 CVE 的列表之后，把所有相关信息存储到数据库中。

3.4 信息融合的策略

第 3.2节中已经分析了各个 API 的异同，本节主要结合 API 的异同介绍如何把不同的 API 的结果整合成一个结果，是主机信息获取的后续。

首先，融合的对象是同一个搜索结果，也就是一个关键词。每个 API 对同一个关键词的搜索结果一定是主机信息的列表，对于每个搜索引擎，这个列表绝大部分情况是不一样的，因此，融合结果的目的即在于把这些主机列表融合，而对于任意一台主机，融合的目标是有用信息尽可能地多。

因为搜索引擎有三个，所以在融合的时候是三者融合。本文采用的方法是两两融合，先融合 Censys 和 Shodan，再把二者融合的结果和 ZoomEye 的结果

融合。

在融合 Censys 和 Shodan 时，可以首先找到二者结果列表中相同 IP 地址的结果，也即相同主机。对于这些交集中的结果，分析每一个字段，把二者相同的字段的进行合并，比如合并列表或者合并字典，而对于不同的字段，直接在融合的结果中把这些字段一并加进去。对于 Censys 中有 Shodan 中没有的主机，再使用一次 Shodan 的/host API 获得这一台主机上的信息，然后把搜索的结果和原来 Censys 的结果进行合并。这么做的原因是可能 Censys 认为这台主机和关键词有关，而 Shodan 的算法认为无关，但是 Shodan 的数据库中实际存放了该主机的信息，在这种情况下，再次使用 Shodan 可以拓宽结果的范围。在实际操作中，该情况确实存在，不过数量不多。对于 Shodan 中有但是 Censys 中没有的主机，考虑到 Censys 提供的信息可以利用的有限，以及 Censys 的访问速度，本文并没有再次用 Censys 的 API 搜索这些主机，而是直接把这些结果加入了二者融合的结果列表。图 3.2是 Censys 和 Shodan 融合的示意图。

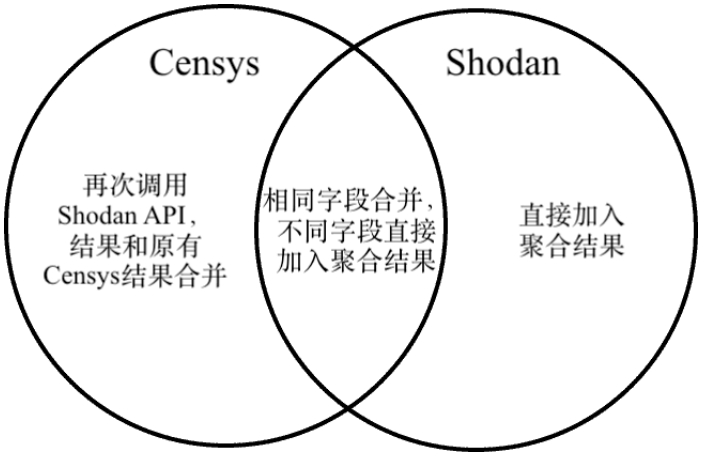

图 3.2　Censys 和 Shodan 搜索结果融合示意图

在融合完 Censys 和 Shodan 之后，得到了一个融合的结果，接下来再用这个结果和 ZoomEye 进行融合。和前述的方法一样，对于相同的 IP 地址的主机，相同字段合并，不同字段直接加入结果，不同 IP 的主机，再去搜索另一方的 API。不过，和上面的一样，考虑到 Censys 的速度和结果，只重新搜索了 Shodan 和 ZoomEye。此外，和上述有一定区别是，对于相同 IP 的主机进行结果合并时，把相同意义但是不同名字的字段只留下了一方的，比如 Shodan 的“location”字段和 ZoomEye 的“geoinfo”字段，由于 Shodan 的“location”字段的信息更加具体，因此在合并时删除了 ZoomEye 的“geoinfo”字段，避免产生其他问题。

此外，在融合的时候，对每个结果添加了一个“source”字段，表示该主机的信息来自哪个 API。比如“Censys/Shodan/ZoomEye”就表示该主机的信息同时来自三个 API，是这三者融合的结果，而如果只有两个，就表示在另外一个 API 里面不存在该主机的信息。这样做可以在出现问题的时候重新去查找该 API 的原始信息，方便开发者定位错误，也方便了用户使用。

在三者的结果融合之后，把融合后的每一台主机的信息存储进入数据库，只有融合后的数据才是存储的对象，而融合前的数据不是。

3.5 分析威胁情报的方法

3.5.1 对比中存在的问题

在获得了详尽的主机信息和 CVE 信息之后，下一步就是对比主机上的信息和 CVE 的信息得到主机上可能存在的威胁。正如前面所述，本文采用的方法是通过对比主机上的开放端口和 CVE 的可能产生威胁的端口，和主机上安装的软件和 CVE 的可能产生威胁的软件。但是，在实际观察了主机上安装的软件的名称和版本和 CVE 的软件名称和版本后，发现软件名称会有多种叫法，比如“Tomcat”和“Apache Tomcat”；版本也会存在多种叫法，比如“1.1.1a”和“1.1.1-log”。因此，如果仅简单使用字符串匹配的方式，很难对比出软件名称和版本是否相同。

此外，不论是对比成功的 CVE 还是由 Shodan 提供的 CVE，其中都有大量的难以利用的 CVE，它们的 CVSS 评分很低，如果也计入系统，会让用户觉得系统里竟然有这么多的漏洞，从而产生困扰。

3.5.2 对比端口和软件的策略

本小节介绍对于一台主机的信息，如何和一个 CVE 的信息进行对比，找到其上的可能的威胁端口和软件。

在对比操作中，首先需要找到安装在主机上的全部的开放端口和全部软件。端口方面，Shodan 提供了“ports”字段，Censys 也提供了“protocol”字段，提取这两个中的信息即可获得全部开放端口。软件方面，遍历 Shodan 提供的“data”字段找到所有服务中运行的软件，和 ZoomEye 中分类好的所有软件信息，即可获得所有的软件信息。

在对比端口的时候，直接把主机开放端口和 CVE 中提供的端口取交集，然后再去除其中的诸如 80 和 443 这样的常见端口即可，因为这样的端口开放只能

证明主机上运行了 HTTP 服务或者 HTTPS 服务，这样的主机非常多，如果判断它们都有可能受到威胁的话是不正确的。相反，有一些不常见的端口却能代表一些软件的使用，比如 4786，是 cisco 的 IOS 软件的默认端口，通过该端口的开放即可判断出该主机上运行了该软件。在这种情况下，可以用该端口开放来判断主机有可能受到该 CVE 的威胁。

接着是判断软件是否存在交集，遍历本机软件和 CVE 中记录的可能的威胁软件。在判断软件是否一样的时候，宗旨是不放过相同的软件但是并不要求完全排除不相同的软件也被当成相同的情况，于是采取一种折中的办法，也就是计分的方法。对软件名称和软件版本分别对比，然后通过一种规则打分，超过一定值就判断两个软件可能一样。在给软件名称打分的时候，应用的方法是求主机软件名称和 CVE 上提供的软件名称的最长公共子串长度占两者平均长度的比例的方法。选用二者平均长度的原因是如果只选取一个长的就会出现比例总是很小的情况，而另一个长的只是在前面加上了供应商的名字；而如果选取短的那个，就会出现如果其中一个软件的名称只有两三个字母直接被另一个包含的情况，这样比例直接为 1，只选择其中一个名字也会有相同的问题，影响了准确性。

最长公共子串是一种基于动态规划的算法，它的目的是寻找两个字符串中最长的公共的子串的算法，这里称为子串的原因是在两个母串中相同的字母必须是相同顺序并且连续。如“abcdefg”和“0bcde123g”的最长子串就是“bcde”，长度为 4，而子序列“bcdeg”因为“g”没有和“bcde”连续，因此并不能算是最长子串。

该算法的思路是，令两个母串分别为 s_1 和 s_2，再令二维数组 c，而 $c[i][j]$ 的含义是母串 s_1 的子串取到母串的第 i 个字符和母串 s_2 的子串取到母串的第 j 个字符时，这两个母串的子串的最长公共子串的长度。因此，如果 $s_1[i] = s_2[j]$，则相当于每个母串的子串都褪掉各自的最后一个字符之后两个新的母串的子串再求最长公共子串的长度加一；如果不等，则各自再尝试褪掉最后一个字符后分别和原来的对方母串求最长公共子串再求二者最大值。于是我们可以得到状态转移方程 3-1：

$$c[i][j] = \begin{cases} 0 & i = 0 \, and \, j = 0 \\ c[i-1][j-1] + 1 & s_1[i] = s_2[j] \\ max(c[i-1][j], c[i][j-1]) & \text{else} \end{cases} \tag{3-1}$$

判断软件版本是否相同方法是，先去掉二者中的非点和数字的字符，然后按照点进行分割，从前往后比较，每次相同则加上一定分数，而越往后分数越高，意味着软件相同的可能性越来越高。

最后，把软件名称的分数和软件版本的分数相加，如果超过一个设定的阈值，则认为两个软件相同。

不过，这里还有一个问题，就是明明不是同一个软件，但是却有相同的软件版本，导致判断软件相同；或者虽然是同一个软件，但因为软件名称完全一样，直接超过了设定的阈值，软件版本相差甚远却认为一样。因此，在首先计算完软件名称的分数之后，如果分数很低，就说明这一定不是相同软件，直接认为两个软件不同；如果分数很高以至于接近 1，就说明很有可能是相同软件，这时把阈值提高，让它再计算软件版本的分数后判断是否超过新的阈值。

对于一台主机来说，如果没有找到任何的可能的威胁端口和软件，则认为该主机不受该 CVE 威胁。

图 3.3是对于一台主机和一个 CVE 进行对比时的流程图。

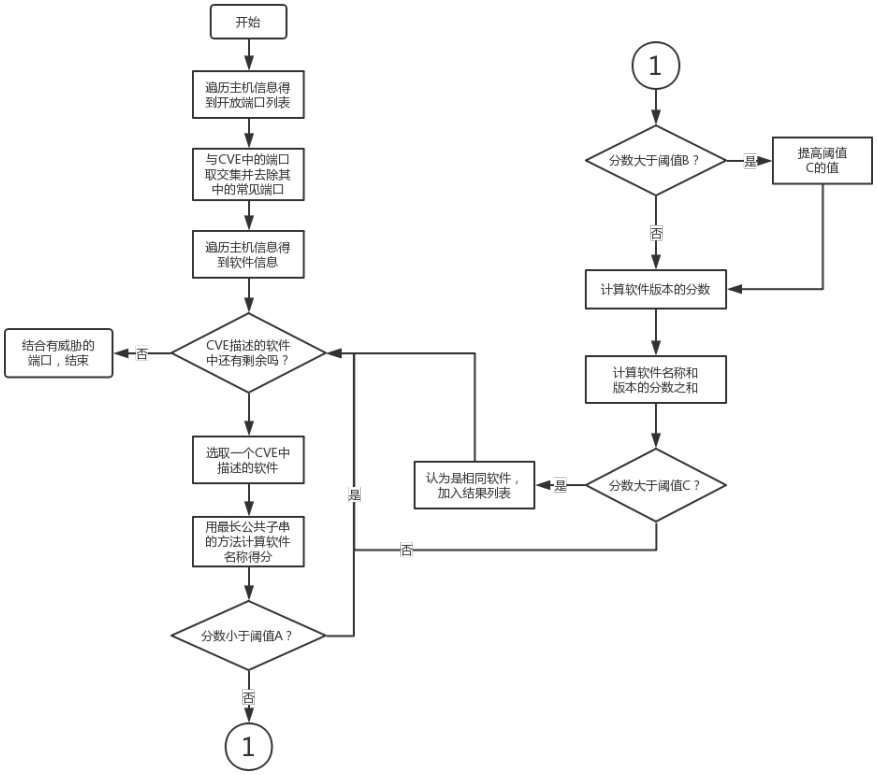

图 3.3　一台主机和一个 CVE 进行对比时的流程图

3.6 获取检测脚本的方法

获取检测脚本的目的是当系统提供给用户某主机存在某可能的威胁时，用户通过阅读 CVE 的介绍并不能清楚地判断该主机是否真的受到这个威胁，这是就需要实际运行脚本来判断。而这种判断方法和渗透测试类似，也具有一定不可预测性，可能对主机产生不良的后果。在整个系统中，处于爬取 CVE 相关信息的步骤，在爬取 CVE Details 网站的时候，也会一并找到检测的脚本。

正如 2.4节中介绍的一样，Exploit Database 是一个漏洞利用脚本的数据库，它也提供了一个网站，展示了各种的利用脚本，而每个脚本都对应了一个 CVE。虽然如此，但是无法直接运行这些脚本来判断主机是否存在该漏洞，因为这些脚本语言不同，运行方法不同，有的甚至存在 bug 无法运行，或者只是一句话让读者自己写代码实现。因此，只需要记录脚本地址然后附到 CVE 信息里即可。

然而，Exploit Database 也没有提供 API，不能直接找到脚本；也不支持爬虫，在爬取该网站的时候有人机检测，无法使用代码直接访问网站。因此，本文使用了第三方搜索引擎来实现找到检测脚本的方法。这里选用的第三方搜索引擎是 Google。在尝试了许多常见的搜索引擎之后，发现只有 Google 能最准确地找到脚本。

具体的方法是在 Google 上搜索“{CVE 名称} site:exploit-db.com”，这样搜索出来的结果就都来自于 Exploit Database。然后再判断 Google 给出的结果是否真的是正确的。一般来说只要再访问结果中提供的网址即可，但是 Exploit Database 不仅限制了非正常访问，连单次代码访问都是禁止的。于是本文读取了 Google 返回的摘要信息，看里面是否存在完整的 CVE 名称，如果存在则结果正确，反之则不然。

然而，在实际操作中发现了一些问题，也即 Google 的访问问题，不论是 Python 上的 Google Search 的模块还是直接用爬虫访问 Google 的网站，亦或用其他代理的 Google，都无法长时间正常地使用，因此在实际系统中并没有加入这一部分，但是代码在该项目的代码仓库中存在。

第 4 章　系统实现

4.1　系统实现与运行环境

本系统的实现采用的编程语言是 Python，具体来说是 Python3，因为 Python 很适合编写服务器以及其语法比较简单，也有丰富的库可以使用，其中就包括三个互联网设备搜索提供的模块，用来处理 API 返回的 json 十分简单。服务器使用的框架是 Flask[11]，因为比较轻量级，功能也比较齐全，也提供了模板引擎，可以大大简化工作。

系统的开发环境是 MacOS，不过可以在安装了相应依赖的任何操作系统上运行。目前本系统在一台 Linux 服务器上运行。

4.2　模块划分

依据第 3章的介绍，本系统共划分为 6 个模块，分别是网站信息抓取模块，具体完成主机信息的抓取；CVE 信息抓取模块，完成 CVE 信息的抓取；数据库操作模块，完成对数据库的读写；推送模块，在完成了威胁情报的分析之后，由该模块进行用户通知的推送；中央控制模块，控制整个系统的运行，包括控制信息抓取的开始与结束，控制信息的存储与读取，威胁情报的分析和最终的推送也在这个模块完成；还有一个网站的部分，仅依赖于系统中的数据库。

在接下来的 6 节里将分别介绍这 6 个模块。

4.3　主机信息抓取模块

因为获得主机信息的工具是互联网搜索引擎的 API，而这些 API 各不相同，肯定要单独写函数，但是，对于很多的关键词抓取，它们的行为又是一样的，也即遍历所有的关键词，对关键词进行检索，而当关键词比较多的时候，对关键词列表的增删操作也是一样的。因此，为了扩展性，将来可以添加更多的互联网搜索引擎 API 进入这个系统，系统首先定义了一个信息抓取的基类 Aggregator，这个基类中一个成员变量表示所有的需要检索的关键词列表，这个列表的每一个成员都是一个 Query 变量，这个类将在下文中介绍。在 Aggregator 类中，实现

了对这个 Query 的列表的操作，如设置列表内容，增加或者删除一个 Query，等等。然后，也实现了对全部的 Query 进行检索的方法，也即遍历列表依次检索并存入一个格式化的返回列表中。而对于每一个 Query 如何进行检索是依 API 不同而异的，因此对一个 Query 进行信息抓取的函数是一个抽象方法，由继承它的各个具体的 Aggregator 来实现。

在使用具体的信息抓取类进行抓取操作时，首先实例化一个 Aggregator 对象，然后设定它的 queries 列表，调用 fetch_all 即可实现抓取操作。

上文中提到的 Query 类，之所以不直接用一个字符串来表示一个检索的目标，是因为一个检索很可能不仅是一个字符串那么简单，它可能有一些其他的限制条件，比如排除其中的一些主机等，而每个 API 的语法都是不同的，如果仅把这些内容用字符串来表示，可能对于每个 API 来说传递给它们的字符串就会不同，这会影响可扩展性。相反，如果在初始化设定检索目标的时候，就把这些信息存入对象，而具体的各个 Aggregator 来实现对这些信息的读取和生成查询 API 时的语句，就能解决这个问题。目前 Query 类只有两个成员变量，一个是表示检索对象的 query，一个是表示对象的类型的 query_type，如表示域名的 hostname、表示 IP 地址段的 net 和表示 IP 地址的 ip。虽然可以由系统自动判断类别，但是目前的实现是在配置文件中由用户指定目标具体属于哪一类。

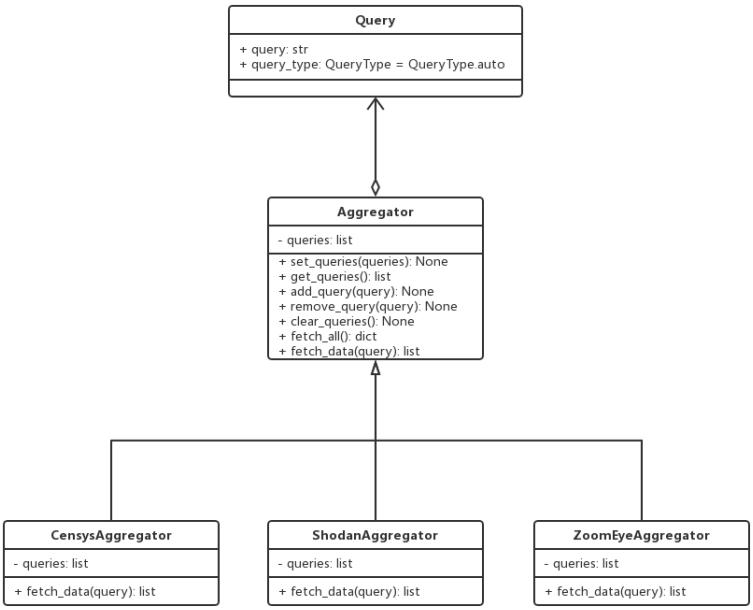

图 4.1　主机信息抓取模块类图

具体的 Aggregator 目前有 3 个，分别是 CensysAggregator，ShodanAggregator 和 ZoomEyeAggregator。这三个类都继承了 Aggregator 类，实现了其中的抽象方法。图 4.1是该模块的类图。从图中可以看出，三个具体的 Aggregator 类和 Aggregator 基类是继承关系，而 Aggregator 和 Query 是聚合关系。

接下来依次介绍每一个 API 的信息抓取的类。

首先是 CensysAggregator。Censys 的 API 的参数中有一个是 page，表示需要结果第几页，一页有 100 个结果，因此第一页就是第 1 至 100 个结果，第二页就是第 101 至第 200 个结果，以此类推。因此在 CensysAggregator 内部又实现了一个方法 fetch_page，可以抓取一页最多 100 条结果的信息。而在返回的结果中，有一个字段“metadata”会记录这些结果的元数据，在“metadata”字段中的“pages”字段就能知道这个关键词的搜索结果会有多少页。所以，首先获得第一页的数据，看到有多少页之后循环获得后面几页的数据后再合并成一个新的列表，再给每个结果加上一个“source”字段赋值为“Censys”即可。

然后是 ShodanAggregator。类似于 Censys，Shodan 的结果也是分页的，但是由于 Shodan 的 API /search 的结果只返回部分数据，需要对每一个结果主机再次进行查询，因此，对每一个主机仅记录其 IP 地址，然后再对每一页的 IP 地址列表进行合并，删除重复的。接着用另一个成员方法 get_info_by_ip 获得其中每一台主机的详细信息，最后加入“source”字段赋值为“Shodan”。其中值得注意的一点是，/host 的 API 是有失败的可能的，因此在实现的时候设置了一个尝试次数，用 try 语句尝试几次，如果都失败了才最终返回一个只含 IP 地址的主机信息。

最后是 ZoomEyeAggregator。ZoomEye 的做法和 Censys 基本一样，也是先获得一页的内容然后再判断有多少页，不过稍微不同的是 ZoomEye 没有直接提供页数的字段，而是总的可用的条目数“available”，需要自己计算页数。在 ZoomEyeAggregator 中还实现了一个方法 get_latest，这个方法是在进行融合的时候使用。因为 ZoomEye 提供数据里面有重复的，因此在融合的时候需要把相同主机的数据里面选出更新时间最近的数据，也即“timestamp”字段最大的数据。

4.4 CVE 信息抓取模块

CVE 信息的抓取虽然也属于信息的抓取，但是和主机信息的抓取的方法和结构完全不一样，所以不需要继承 Aggregator 类，而是定义了一个新的类 CVEAggregator，专门用于 CVE 信息的抓取。

类似于 Aggregator 的设计，在 CVEAggregator 中也加入了一个私有变量 cves，用于存放需要获取信息的 CVE 列表。在实际使用中，类似地，先实例化一个对象，设定该列表之后调用 update_cves 方法来更新列表中的 CVE。而针对整个列表，也实现了一些相关的增删操作。

然而，在开始获取 CVE 信息前，大部分情况下是不知道 CVE 的名称的，如果想要获得所有的 CVE 信息，或者某几年的 CVE 信息，需要首先得到这些 CVE 的名字，才能访问网页来进行爬虫的工作。因此，内部还实现了一个 get_all_cves 方法用于获得目前所有的 CVE 的名字的列表和一个 get_cves_by_years 方法用于获得某几年的 CVE 的名字的列表。其中后一个方法依赖于前一个方法，也即获得某几年的列表的时候，是首先获得全部的 CVE 列表再挑出来哪些是这几年的的。而获得所有 CVE 的名字的列表的方法也是去访问一个链接，这个链接以纯文本的格式记录了所有的 CVE 的基本信息，可以用爬虫的方法获得其中的 CVE 名称。

而对于任意一个 CVE，爬取其信息的方法就是标准的爬虫方法，先用 CVE 的名称得到对应的 CVE 的信息的页面，然后用 requests 库得到其 HTML，接着用 BeautifulSoup 解析该页面找到相应的数据即可。不过，有一些在前述方法中得到的 CVE 名称在 CVE Details 网站里不存在，需要提前判断排除。此外，有的网站存在不规范的现象，比如内含非法字符“&#”，后面跟的内容不是数字无法构成字符，需要用正则表达式匹配去除该字符。

4.5 数据库操作模块

在本系统中，有主机信息、CVE 信息和威胁情报信息都需要进行存取的操作，其中主机信息是从 API 中获得的，在未经加工和融合之后的情况下是字典，很容易格式化为 json，而且内部存在非常复杂的字典和列表的嵌套关系；而 CVE 信息是系统抓取的，也存为字典；威胁情报也是系统分析的，在内存中存也为字典。对于这种字典和列表有复杂嵌套关系的数据，非常容易选择数据库的类型为非关系型数据库，因此系统中选择的数据库为 MongoDB[12]，而且 Python 中也有 pymongo 模块，可以轻松地操作 MongoDB 中的增删改查。

实现的时候，采用的方法是类似于 namespace 的方法定义个了一个 MongoHelper 类，实际上这个类并没有成员变量也没有成员方法，都是静态方法，这么做的目的是不希望有数据库读写的方法散落在各个文件中，希望可以集中所有

的数据库操作的方法到一起，也能方便修改和调试。

在本系统中，共有 1 个数据库和三个 Collection，分别存储主机的信息、CVE 的信息和分析结果的威胁情报信息。在该类中，所有的方法都是围绕这三个 Collection 进行增删改查，比如存储一系列主机信息的方法、根据 IP 地址找到一台主机的信息的方法、根据 CVE 名字获得威胁情报的方法等。存储的方法主要用于数据抓取的过程，而读取的方法主要用于网站的实现，在网站中，为了给用户呈现尽量多的数据和用户体验，实现了很多根据某一些条件查询另一些数据的方法。

4.6 推送模块

推送服务是在一次抓取信息和分析威胁结束之后做的，由系统发送邮件给用户，通知本次分析的结果，比如共多少台主机存在可能的高危漏洞，用户关注的 CVE 中各有多少台主机可能受到威胁等。

这里采用的发送邮件的方法是使用 SMTP 协议，用 Python 的 smtplib 模块。在一次扫描结束后，建立一个 smtplib.SMTP 对象，填写服务器地址和端口两个参数，然后用格式化的数据编辑好通知内容，发送给用户的之前设定好的邮箱。但是，用这种方法很容易产生的一个问题就是会被放入垃圾邮件甚至直接删除，这里就需要用户主动添加发件人的邮箱至白名单了。因为用 SMTP 发送邮件是最简单的做法，而正因为如此目前大部分邮箱服务都提供了过滤的功能，避免过多垃圾邮件。

4.7 中控模块

4.7.1 控制信息获取功能

中控模块是本系统最核心的一个模块，第 3.1.2节中的系统流程图中的所有步骤都是该模块所做，而其所做工作中除了初始化外第一步就是控制各个 Aggregator 抓取网络中的主机和 CVE 的信息。

在中控类 Controller 中存有 queries 和 cves 两个成员变量，分别存储当前本系统需要获取主机的信息的 Query 列表和 CVE 列表。在 Controller 的主要函数 start_aggregate 中，首先初始化这两个变量，然后开始主机信息的抓取和 CVE 信息的抓取。

在主机信息的抓取中，首先初始化三个 Aggregator 的对象，然后设定它们各自的 queries 为 Controller 的 queries，接着就调用 fetch_all 开始各自的抓取工作。在各自的工作都完成之后，就开始融合三个数据源的主机信息，使用的方法即第 3.4节中介绍的方法。不过，此处稍有不同的是，在最后一步存储数据进入数据库前，需要对融合的数据进行进一步的处理，因为 MongoDB 对存储的数据有要求。比如 MongoDB 最大存储 64 位整数，而有一个“serial”字段的值可能会超过 10^{30}，此时就需要把这个值转换成字符串再存入数据库。又如 MongoDB 要求字典的键中不能含有“.”，而比如“metadata.os”这个键就含有，异或其他更加深层的键中也有。此处采用递归的方法对所有的键进行遍历，把点换成了下划线。诸如此类的特殊处理是在实际操作中一个一个排查的，可能在 API 信息的扩充中需要增加新的特殊处理。

而在 CVE 信息的抓取中，同样是初始化一个 CVEAggregator 对象然后初始化抓取目标后进行抓取，但是为了提高效率，此处采用了多线程的方法让同时有多个对象在进行信息的抓取。每个线程中都有一个抓取用的对象，然后在初始化的时候去访问共享变量 cves，取其中的前定值个 CVE（目前设定为 100），然后开始这些 CVE 的详情抓取。为了解决同步互斥的问题，在访问全部 CVE 列表前需要获得锁。在一个线程完成自己的一小部分工作后，它还回去重新访问这个列表看看任务有没有被拿空，如果没有则继续拿若干个，空了则该线程结束工作。因为抓取的 CVE 的之间各个都没有依赖关系，所以可以用这种方式提高效率。

4.7.2 分析威胁情报功能

在抓取完主机和 CVE 信息之后就需要进行分析威胁情报了。

在进行分析的时候采取的方法和第 3.5.2节中介绍的方法完全一致，只是为了提高效率同样采用了多线程的处理方法。如果设主机的数量为 n，CVE 的数量为 m，为了对每一台主机遍历全部的 CVE，时间复杂度至少为 $O(mn)$，这是一个非常长的时间，而对每一台主机进行分析之间并没有依赖关系，所以可以采用多线程的方法节省时间。此处的方法同样类似 CVE 处理的时候的处理方法，每个线程都在开始任务前取共享变量的主机列表中的若干项，然后处理完成后重新访问再去访问直到取空。

4.7.3 配置文件

在第 4.7.1节中提到 Controller 类中有两个成员变量 queries 和 cves，为了设置这两个变量的值，不是在代码中设置，而是使用配置文件 config.json 来设置这两个变量。在配置文件中，不仅能够设置需要监测的主机范围和 CVE 范围，还能设置推送的时候推送的邮箱列表和着重关注的 CVE 列表。

在这个文件中，任何字段都有其固定的格式，比如“CVEs”这个字段里面的二级字段“years”可以设定 CVE 的年份，“others”可以添加一些其他的 CVE，“all”为 True 表示监测全部 CVE。

在初始化的时候，由 Controller 来解析这个文件并填入这两个变量，之后的操作中都可以用成员变量来访问就不需要再访问文件了。

4.7.4 定时功能

为了让系统具有实时性，并且各个 API 也是定时更新的，CVE Details 和全部 CVE 的列表也是会更新的，定时更新整个数据库就变得很有必要。此处定时采用的方法比较简单，即把 Controller 运行在另一个进程中，而非另一个线程（为了避免 GIL 的问题），然后在 Controller 完成一次 start_aggregate 之后，调用 time.sleep，推迟线程的运行，在到达设定时间之后，Controller 会再次运行。目前设定的时间周期是一周，也即公式 4-1

$$7 \times 24 \times 60 \times 60 = 604800\,sec \tag{4-1}$$

将该参数填入 sleep 的参数中即可。

在启动系统的时候，需要首先启动 Controller，然后再启动服务器。

4.7.5 中控模块类图

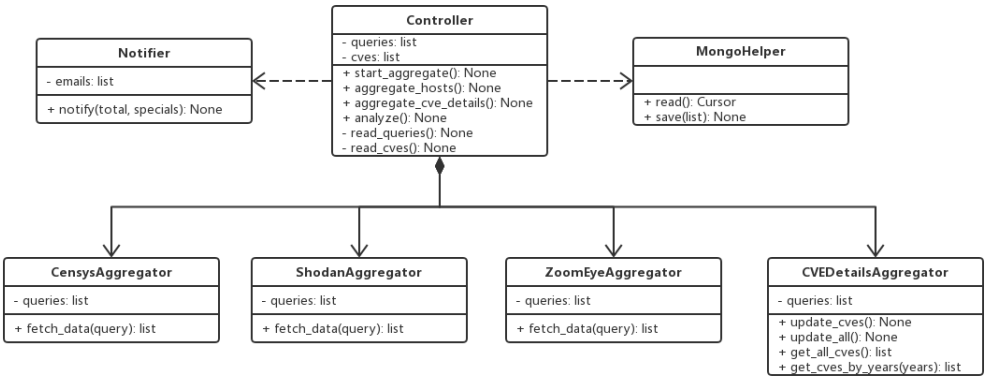

图 4.2　Controller 的类图

图 4.2是 Controller 的类图。它描述了 Controller 类和之前描述的几个模块之前的关系，也即和各个 Aggregator 之间是组合关系，而和 Notifier 和 MongoHelper 之间是依赖关系。

4.8　网站

网站的模块是和主要逻辑无关的一个模块，为了让用户能更加简单地使用该系统，于是制作了一个简单的 Flask 框架的 web 网站。

网站总共分为/index 展示有可能有高危漏洞的主机列表；/details 展示某一台主机的详情，包括基本信息、高危 CVE 信息、开放端口和运行的服务等；/search 用来展示搜索的结果；/raw 用来展示一台主机的 json 格式的详细信息的原始内容，共 4 个路径。

不论是哪个路径，基本的思路都是通过某些信息去查数据库，然后再通过 jinja2 模板引擎展示在页面上。

第 5 章　结果检验

5.1　网站介绍

5.1.1　展示全部可能有漏洞的主机功能

本系统最终的呈现方式是一个网站，具有展示可能具有漏洞的主机和相应的搜索功能。图 5.1是网站的首页，也是能够展示全部可能具有高危漏洞的主机的地方。在此例中，监测的主机范围是 166.111.0.0/16，监测的 CVE 范围是全部 2018 年的 CVE。

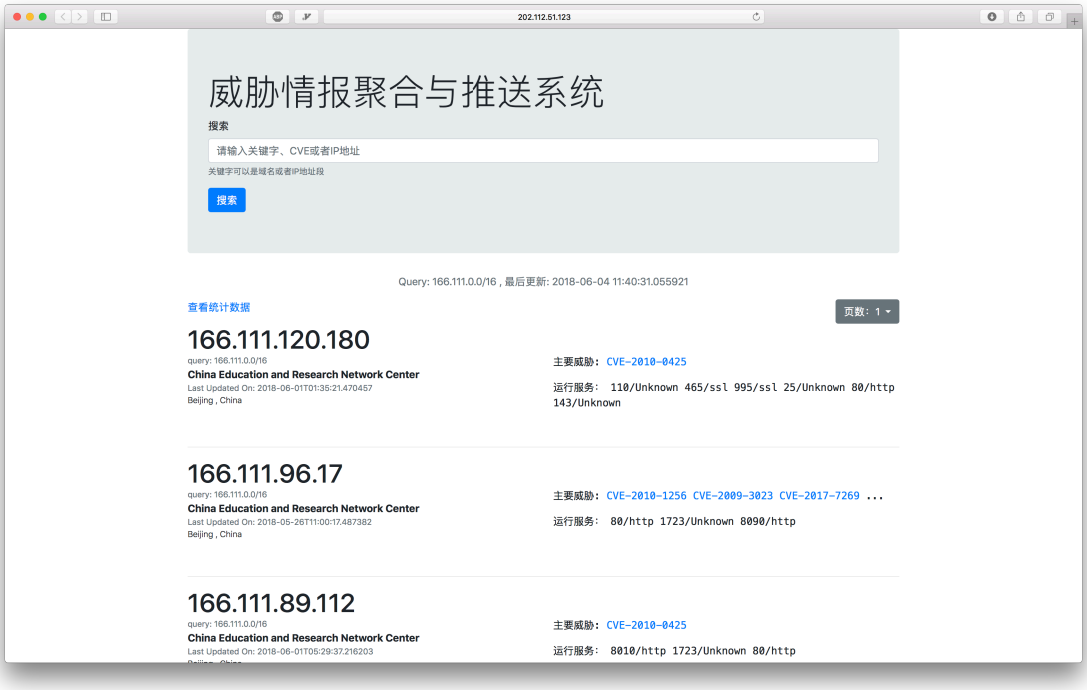

图 5.1　网站主页

可以看到，网站的主要部分是一个列表，这列表里面的主机都可能存在高危漏洞，也即 CVSS 评分至少为 8 的 CVE。每一个项目的左侧是主机的基本信息，右侧显示了主要的威胁和运行的服务。点击 IP 地址可以进入主机的详情页面，点击 CVE 可以进入该 CVE 的详情页面。

在右侧有一个能够进行页面跳转的按钮，可以选择当前显示第几页的内容，

因为 1 页最多显示 10 条。

5.1.2 主机详情功能

在点击主页上的主机的 IP 地址之后可以进入该主机的详情界面，图 5.2展示了其中一台 IP 地址为 166.111.68.30 主机的详情。

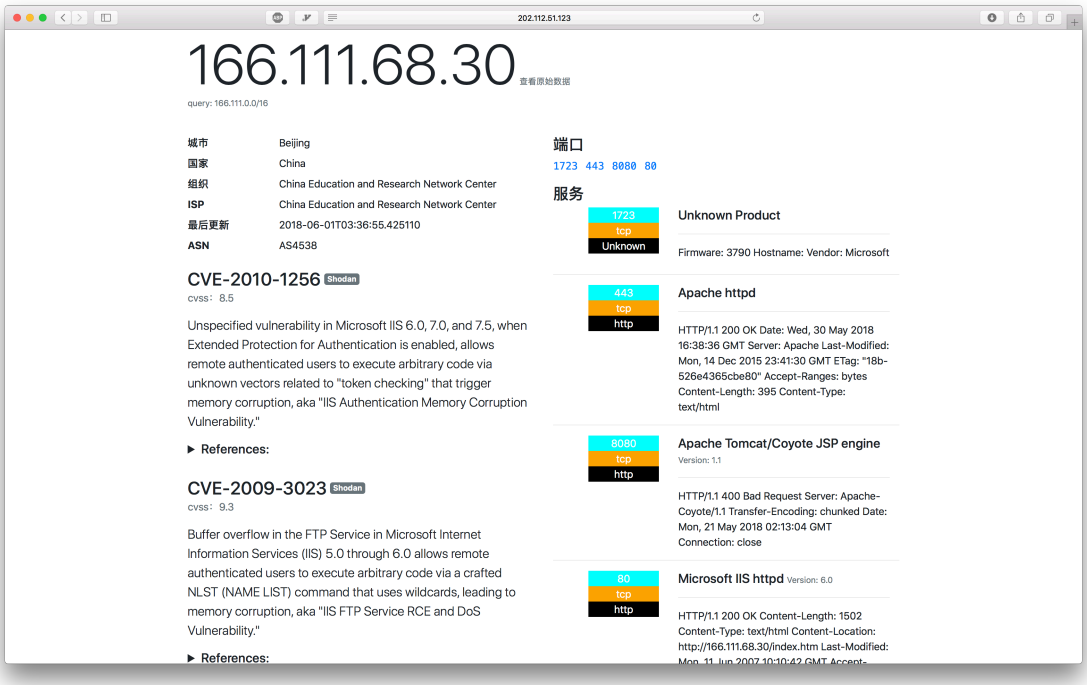

图 5.2 主机详情界面

在该界面，可以看到该主机的基本信息，可能受到威胁的 CVE 以及对应的详细信息，这里提供了简介、CVSS 评分、相关阅读等功能，还能看到该主机开放的端口和运行的服务和其上运行的软件。点击 IP 地址旁边的查看原始数据还能看到原始 json 格式的主机信息。

5.1.3 搜索功能

在首页的搜索栏里输入相应关键词可以进行一些简单的搜索，比如搜索 IP 地址、CVE 名称或者 Query 的名称。图 5.3是搜索关键词“CVE-2018-0171”的结果，显示了一台可能受到该 CVE 威胁的主机。

在图中可以看出，该主机的 IP 地址为 166.111.176.55，是监测的范围 166.111.0.0/16 中的一台，右侧的 CVE 被染红表示这是关键词。

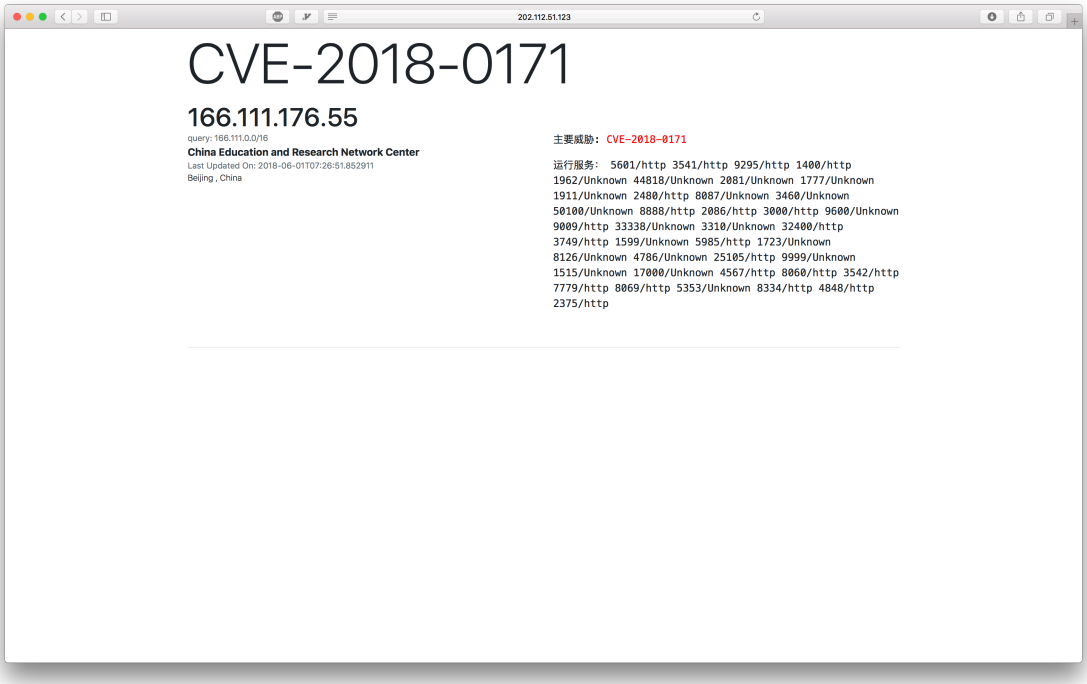

图 5.3　CVE-2018-0171 的搜索结果

作为对比，在 Shodan 中搜索该 CVE，没有结果，表明在此例中系统的聚合和分析功能已经比 Shodan 具有更多的数据和更强的分析能力。

5.1.4　统计功能

点击在首页的左侧的“查看统计数据”可以看到的监测的主机范围内的一些统计数据。图 5.4是统计数据的页面。

在页面中我们可以看到在 166.111.0.0/16 这个范围内开放最多的端口中的 5 个、运行最多的协议中的 5 个、可能受到威胁的 CVE 最多的 5 个。值得注意的一点是，这里的统计数据是根据来自 3 个数据源聚合之后的结果，因此比单个数据源 API 要更加准确，这是本系统的优势之一。

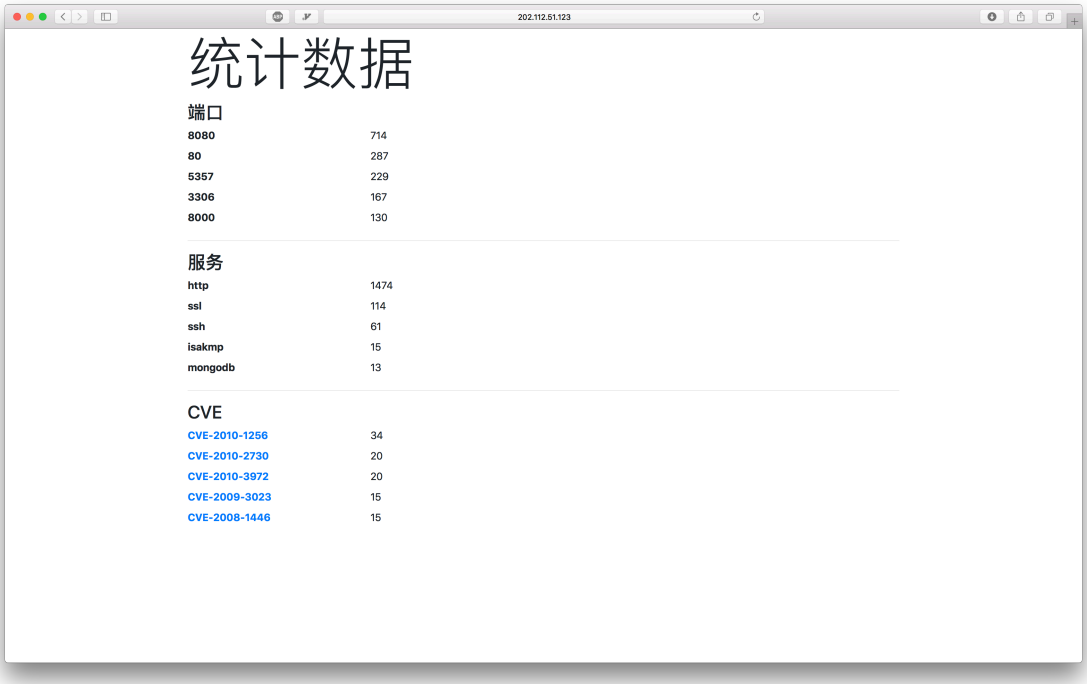

图 5.4　统计数据页面

5.2　通知

本系统的另一个主要特点就是具有推送的功能，这是目前几个搜索引擎都不具备的功能。图 5.5就是该系统推送后邮箱收到的结果。

图 5.5　系统推送分析结果

5.3　举例验证

CVE-2018-0171 是一个 Cisco 发布的利用 Smart Install 的代码漏洞的远程代码执行严重漏洞，攻击者可以无需验证就向远端 Cisco 设备的 4786 端口发送恶意数据包。采用本系统对该漏洞进行检验。在图 5.3中我们就能看到搜索该 CVE 得到的结果，发现有一台主机可能受到该漏洞的威胁。进入该主机的原始信息

里可以看到该主机的“_shodan”字段的“module”字段为 cisco-smi，确实和 cisco 的 IOS 软件相关，且开放了 4786 端口，该主机的确受到了该 CVE 的威胁，系统做出的行为正确。图 5.6即为显示系统显示的界面，点击“查看原始数据”即可找到上述 cisco-msi，验证是否真的受到威胁，或者进一步运行检测脚本进行实际检验。

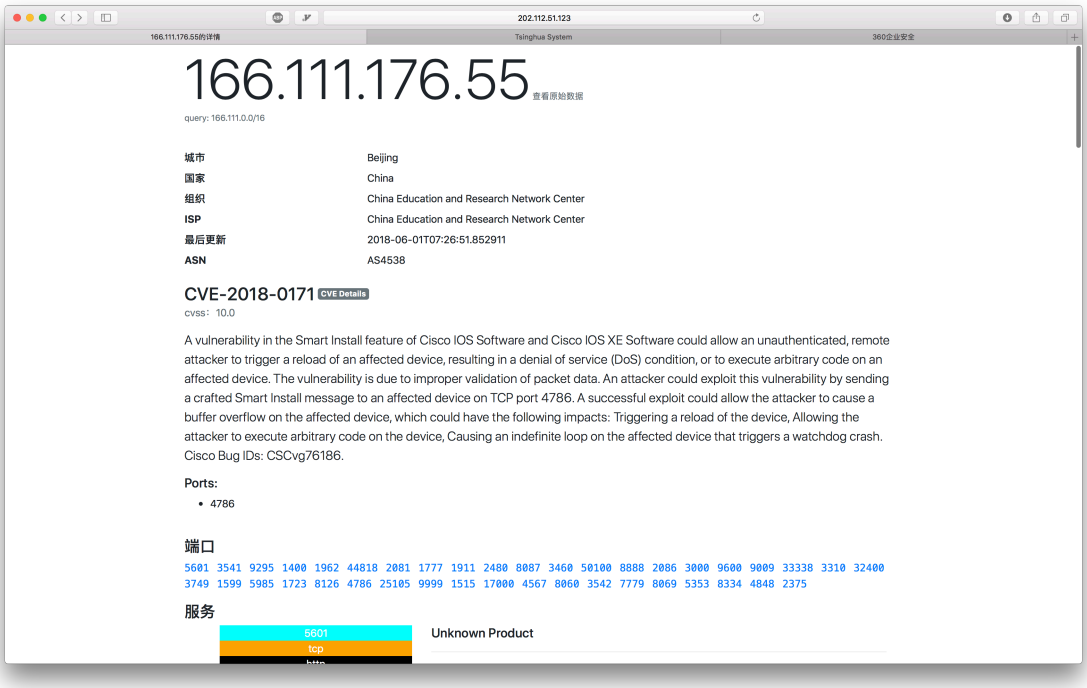

图 5.6　可能受到 CVE-2018-0171 的主机

第 6 章　总结与展望

本文设计并实现了一个威胁情报融合与推送系统，该系统能对市面上能见到的最主流的三个互联网设备搜索引擎进行数据的融合，用这些数据和 CVE 的数据进行对比，然后找到主机中可能的高危漏洞。补足了现有的网络漏洞扫描工具漏洞库更新不及时，不能扫描未授权主机，自动化程度低等问题。

不过，本系统仍然存在一些不足。比如，抓取的速度不够快，在进行数据融合的时候三个数据源实际没有依赖关系，但是并未采取并行的方法；又如，主机数据和 CVE 信息进行比对的时候的算法没有任何先验知识，如果能运用一些先验知识或者机器学习的方法进行软件是否相同的判断可能会有更加准确的结果；此外，现在对比主机和 CVE 的方法仅是通过软件和端口，并未使用其他指纹信息，如果能加入 Shodan 和 Censys 提供的更加详细的指纹信息，运用自然语言处理的方法结合 CVE 的描述，会有更好的比对效果。

插图索引

表格索引

公式索引

参考文献

[1] Ehrenfeld J M. Wannacry, cybersecurity and health information technology: A time to act[J]. Journal of medical systems, 2017, 41(7): 104.

[2] O'Gorman J, Kearns D, Aharoni M. Metasploit: the penetration tester's guide[M]. [S.l.]: No Starch Press, 2011

[3] Damele B, Stampar M. Sqlmap[J]. Online at http://sqlmap. org, 2012.

[4] Thacker B H, Riha D S, Fitch S H, et al. Probabilistic engineering analysis using the nessus software[J]. Structural Safety, 2006, 28(1-2): 83-107.

[5] Lyon G F. Nmap network scanning: The official nmap project guide to network discovery and security scanning[M]. [S.l.]: Insecure, 2009

[6] Durumeric Z, Adrian D, Mirian A, et al. A search engine backed by internet-wide scanning [C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. [S.l.]: ACM, 2015: 542-553.

[7] Wiemer S. A software package to analyze seismicity: Zmap[J]. Seismological Research Letters, 2001, 72(3): 373-382.

[8] Matherly J C. Shodan the computer search engine[J]. Available at [Online]: http://www. shodanhq. com/help, 2009.

[9] Singh S. Breaking the sandbox. exploit-db. com[Z]. [S.l.: s.n.], 2015.

[10] Mell P, Scarfone K, Romanosky S. A complete guide to the common vulnerability scoring system version 2.0[C]//Published by FIRST-Forum of Incident Response and Security Teams: volume 1. [S.l.: s.n.], 2007: 23.

[11] Grinberg M. Flask web development: developing web applications with python[M]. [S.l.]: " O'Reilly Media, Inc.", 2018

[12] Chodorow K. Mongodb: The definitive guide: Powerful and scalable data storage[M]. [S.l.]: " O'Reilly Media, Inc.", 2013

[13] 薛瑞尼. ThuThesis: 清华大学学位论文模板[EB/OL]. 2017. https://github.com/xueruini/thuthesis.

致　谢

衷心感谢导师尹霞教授和段海新教授对本人的精心指导。他们的在项目选题、项目开发、答辩和论文书写中为我提出了宝贵的建议，使我受益匪浅。

网研院的张甲老师在组会和私下都对我的项目进度十分关心，提出了很多重要的修改建议，没有他，我也无法完成毕业设计，感谢他的帮助与支持。

感谢王奥丞同学和靳子豪同学在完成项目过程中为我解答了许多细节问题，从代码书写到思路整理，他们都为本文做出了贡献，令我十分感动，感谢他们。

感谢 ProcessOn，帮助我在绘制美观的插图过程中节省了大量的时间。

感谢 LaTeX 和 ThuThesis[13]，帮我节省了极多的排版的时间。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　外文资料原文

The title of the English paper

Abstract: As one of the most widely used techniques in operations research, *mathematical programming* is defined as a means of maximizing a quantity known as *bjective function*, subject to a set of constraints represented by equations and inequalities. Some known subtopics of mathematical programming are linear programming, nonlinear programming, multiobjective programming, goal programming, dynamic programming, and multilevel programming[1].

It is impossible to cover in a single chapter every concept of mathematical programming. This chapter introduces only the basic concepts and techniques of mathematical programming such that readers gain an understanding of them throughout the book[2,3].

A.1　Single-Objective Programming

The general form of single-objective programming (SOP) is written as follows,

$$\begin{cases} \max \ \ f(x) \\ \text{subject to:} \\ \qquad g_j(x) \le 0, \quad j = 1, 2, \cdots, p \end{cases} \tag{123}$$

which maximizes a real-valued function f of $x = (x_1, x_2, \cdots, x_n)$ subject to a set of constraints.

Definition A.1： In SOP, we call x a decision vector, and $x_1, x_2, \cdots, x_n$ decision variables. The function f is called the objective function. The set

$$S = \left\{ x \in \mathfrak{R}^n \;\middle|\; g_j(x) \le 0,\ j = 1, 2, \cdots, p \right\} \tag{456}$$

is called the feasible set. An element x in S is called a feasible solution.

Definition A.2: A feasible solution x^* is called the optimal solution of SOP if and only if

$$f(x^*) \geq f(x) \tag{A-1}$$

for any feasible solution x.

One of the outstanding contributions to mathematical programming was known as the Kuhn-Tucker conditionsA-2. In order to introduce them, let us give some definitions. An inequality constraint $g_j(x) \leq 0$ is said to be active at a point x^* if $g_j(x^*) = 0$. A point x^* satisfying $g_j(x^*) \leq 0$ is said to be regular if the gradient vectors $\nabla g_j(x)$ of all active constraints are linearly independent.

Let x^* be a regular point of the constraints of SOP and assume that all the functions $f(x)$ and $g_j(x), j = 1, 2, \cdots, p$ are differentiable. If x^* is a local optimal solution, then there exist Lagrange multipliers $\lambda_j, j = 1, 2, \cdots, p$ such that the following Kuhn-Tucker conditions hold,

$$\begin{cases} \nabla f(x^*) - \sum_{j=1}^{p} \lambda_j \nabla g_j(x^*) = 0 \\ \lambda_j g_j(x^*) = 0, \quad j = 1, 2, \cdots, p \\ \lambda_j \geq 0, \quad j = 1, 2, \cdots, p. \end{cases} \tag{A-2}$$

If all the functions $f(x)$ and $g_j(x), j = 1, 2, \cdots, p$ are convex and differentiable, and the point x^* satisfies the Kuhn-Tucker conditions (A-2), then it has been proved that the point x^* is a global optimal solution of SOP.

A.1.1 Linear Programming

If the functions $f(x), g_j(x), j = 1, 2, \cdots, p$ are all linear, then SOP is called a *linear programming*.

The feasible set of linear is always convex. A point x is called an extreme point of convex set S if $x \in S$ and x cannot be expressed as a convex combination of two points in S. It has been shown that the optimal solution to linear programming corresponds to an extreme point of its feasible set provided that the feasible set S is bounded. This fact is the basis of the *simplex algorithm* which was developed by Dantzig as a very efficient method for solving linear programming.

Table 1 This is an example for manually numbered table, which would not appear in the list of tables

Network Topology		# of nodes	# of clients			Server
GT-ITM	Waxman Transit-Stub	600	2%	10%	50%	Max. Connectivity
Inet-2.1		6000				
Xue	Rui	Ni	ThuThesis			
	ABCDEF					

Roughly speaking, the simplex algorithm examines only the extreme points of the feasible set, rather than all feasible points. At first, the simplex algorithm selects an extreme point as the initial point. The successive extreme point is selected so as to improve the objective function value. The procedure is repeated until no improvement in objective function value can be made. The last extreme point is the optimal solution.

A.1.2 Nonlinear Programming

If at least one of the functions $f(x), g_j(x), j = 1, 2, \cdots, p$ is nonlinear, then SOP is called a *nonlinear programming*.

A large number of classical optimization methods have been developed to treat special-structural nonlinear programming based on the mathematical theory concerned with analyzing the structure of problems.

Figure 1 This is an example for manually numbered figure, which would not appear in the list of figures

Now we consider a nonlinear programming which is confronted solely with maximizing a real-valued function with domain $\mathfrak{R}^n$. Whether derivatives are available or not, the usual strategy is first to select a point in $\mathfrak{R}^n$ which is thought to be the most likely place where the maximum exists. If there is no information available on which to base such a selection, a point is chosen at random. From this first point an attempt is made to construct a sequence of points, each of which yields an improved objective function value over its predecessor. The next point to be added to the sequence is cho-

sen by analyzing the behavior of the function at the previous points. This construction continues until some termination criterion is met. Methods based upon this strategy are called *ascent methods*, which can be classified as *direct methods*, *gradient methods*, and *Hessian methods* according to the information about the behavior of objective function f. Direct methods require only that the function can be evaluated at each point. Gradient methods require the evaluation of first derivatives of f. Hessian methods require the evaluation of second derivatives. In fact, there is no superior method for all problems. The efficiency of a method is very much dependent upon the objective function.

A.1.3 Integer Programming

Integer programming is a special mathematical programming in which all of the variables are assumed to be only integer values. When there are not only integer variables but also conventional continuous variables, we call it *mixed integer programming*. If all the variables are assumed either 0 or 1, then the problem is termed a *zero-one programming*. Although integer programming can be solved by an *exhaustive enumeration* theoretically, it is impractical to solve realistically sized integer programming problems. The most successful algorithm so far found to solve integer programming is called the *branch-and-bound enumeration* developed by Balas (1965) and Dakin (1965). The other technique to integer programming is the *cutting plane method* developed by Gomory (1959).

Uncertain Programming (*BaoDing Liu,* 2006.2)

References

NOTE: These references are only for demonstration. They are not real citations in the original text.

[1] Donald E. Knuth. The TEXbook. Addison-Wesley, 1984. ISBN: 0-201-13448-9

[2] Paul W. Abrahams, Karl Berry and Kathryn A. Hargreaves. TEX for the Impatient. Addison-Wesley, 1990. ISBN: 0-201-51375-7

[3] David Salomon. The advanced TEXbook. New York : Springer, 1995. ISBN:0-387-94556-3

附录 B　外文资料的调研阅读报告或书面翻译

英文资料的中文标题

摘要：本章为外文资料翻译内容。如果有摘要可以直接写上来，这部分好像没有明确的规定。

B.1　单目标规划

北冥有鱼，其名为鲲。鲲之大，不知其几千里也。化而为鸟，其名为鹏。鹏之背，不知其几千里也。怒而飞，其翼若垂天之云。是鸟也，海运则将徙于南冥。南冥者，天池也。

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \tag{123}$$

吾生也有涯，而知也无涯。以有涯随无涯，殆已！已而为知者，殆而已矣！为善无近名，为恶无近刑，缘督以为经，可以保身，可以全生，可以养亲，可以尽年。

B.1.1　线性规划

庖丁为文惠君解牛，手之所触，肩之所倚，足之所履，膝之所倚，砉然响然，奏刀騞然，莫不中音，合于桑林之舞，乃中经首之会。

表 1　这是手动编号但不出现在索引中的一个表格例子

Network Topology		# of nodes	# of clients			Server
GT-ITM	Waxman Transit-Stub	600	2%	10%	50%	Max. Connectivity
Inet-2.1		6000				
Xue	Rui	Ni	ThuThesis			
	ABCDEF					

文惠君曰：“嘻，善哉！技盖至此乎？”庖丁释刀对曰：“臣之所好者道也，进乎技矣。始臣之解牛之时，所见无非全牛者；三年之后，未尝见全牛也；方今之时，臣以神遇而不以目视，官知止而神欲行。依乎天理，批大郤，导大窾，因其

固然。技经肯綮之未尝，而况大軱乎！良庖岁更刀，割也；族庖月更刀，折也；今臣之刀十九年矣，所解数千牛矣，而刀刃若新发于硎。彼节者有间而刀刃者无厚，以无厚入有间，恢恢乎其于游刃必有余地矣。是以十九年而刀刃若新发于硎。虽然，每至于族，吾见其难为，怵然为戒，视为止，行为迟，动刀甚微，謋然已解，如土委地。提刀而立，为之而四顾，为之踌躇满志，善刀而藏之。”

文惠君曰：“善哉！吾闻庖丁之言，得养生焉。”

B.1.2 非线性规划

孔子与柳下季为友，柳下季之弟名曰盗跖。盗跖从卒九千人，横行天下，侵暴诸侯。穴室枢户，驱人牛马，取人妇女。贪得忘亲，不顾父母兄弟，不祭先祖。所过之邑，大国守城，小国入保，万民苦之。孔子谓柳下季曰：“夫为人父者，必能诏其子；为人兄者，必能教其弟。若父不能诏其子，兄不能教其弟，则无贵父子兄弟之亲矣。今先生，世之才士也，弟为盗跖，为天下害，而弗能教也，丘窃为先生羞之。丘请为先生往说之。”

图 1 这是手动编号但不出现索引中的图片的例子

柳下季曰：“先生言为人父者必能诏其子，为人兄者必能教其弟，若子不听父之诏，弟不受兄之教，虽今先生之辩，将奈之何哉？且跖之为人也，心如涌泉，意如飘风，强足以距敌，辩足以饰非。顺其心则喜，逆其心则怒，易辱人以言。先生必无往。”

孔子不听，颜回为驭，子贡为右，往见盗跖。

B.1.3 整数规划

盗跖乃方休卒徒大山之阳，脍人肝而餔之。孔子下车而前，见谒者曰：“鲁人孔丘，闻将军高义，敬再拜谒者。”谒者入通。盗跖闻之大怒，目如明星，发上指冠，曰：“此夫鲁国之巧伪人孔丘非邪？为我告之：尔作言造语，妄称文、武，冠枝木之冠，带死牛之胁，多辞缪说，不耕而食，不织而衣，摇唇鼓舌，擅生是非，以迷天下之主，使天下学士不反其本，妄作孝弟，而侥幸于封侯富贵者也。子之罪大极重，疾走归！不然，我将以子肝益昼餔之膳。”

附录 C　其它附录

前面两个附录主要是给本科生做例子。其它附录的内容可以放到这里，当然如果你愿意，可以把这部分也放到独立的文件中，然后将其 `\input` 到主文件中。

在学期间参加课题的研究成果

个人简历

xxxx 年 xx 月 xx 日出生于 xx 省 xx 县。

xxxx 年 9 月考入 xx 大学 xx 系 xx 专业，xxxx 年 7 月本科毕业并获得 xx 学士学位。

xxxx 年 9 月免试进入 xx 大学 xx 系攻读 xx 学位至今。

发表的学术论文

[1] Yang Y, Ren T L, Zhang L T, et al. Miniature microphone with silicon- based ferroelectric thin films. Integrated Ferroelectrics, 2003, 52:229-235. (SCI 收录, 检索号:758FZ.)

[2] 杨轶, 张宁欣, 任天令, 等. 硅基铁电微声学器件中薄膜残余应力的研究. 中国机械工程, 2005, 16(14):1289-1291. (EI 收录, 检索号:0534931 2907.)

[3] 杨轶, 张宁欣, 任天令, 等. 集成铁电器件中的关键工艺研究. 仪器仪表学报, 2003, 24(S4):192-193. (EI 源刊.)

[4] Yang Y, Ren T L, Zhu Y P, et al. PMUTs for handwriting recognition. In press. (已被 Integrated Ferroelectrics 录用. SCI 源刊.)

[5] Wu X M, Yang Y, Cai J, et al. Measurements of ferroelectric MEMS microphones. Integrated Ferroelectrics, 2005, 69:417-429. (SCI 收录, 检索号:896KM)

[6] 贾泽, 杨轶, 陈兢, 等. 用于压电和电容微麦克风的体硅腐蚀相关研究. 压电与声光, 2006, 28(1):117-119. (EI 收录, 检索号:06129773469)

[7] 伍晓明, 杨轶, 张宁欣, 等. 基于 MEMS 技术的集成铁电硅微麦克风. 中国集成电路, 2003, 53:59-61.

研究成果

[1] 任天令, 杨轶, 朱一平, 等. 硅基铁电微声学传感器畴极化区域控制和电极连接的方法: 中国, CN1602118A. (中国专利公开号)

[2] Ren T L, Yang Y, Zhu Y P, et al. Piezoelectric micro acoustic sensor based on ferroelectric materials: USA, No.11/215, 102. (美国发明专利申请号)

综合论文训练记录表

学生姓名		学号		班级	
论文题目					
主要内容以及进度安排	指导教师签字： 考核组组长签字： 年　　月　　日				
中期考核意见	考核组组长签字： 年　　月　　日				

指导教师评语	指导教师签字：_____ 年　　月　　日
评阅教师评语	评阅教师签字：_____ 年　　月　　日
答辩小组评语	答辩小组组长签字：_____ 年　　月　　日

总成绩：_____

教学负责人签字：_____

年　　月　　日