**分类号：TP311.1 U D C：D10621-408-(2020)3181-0**

**密 级：公 开 编 号：2016121035**

**成都信息工程大学**

**学位论文**

**基于机器学习的恶意代码检测软件的开发**

|  |  |
| --- | --- |
| **论文作者姓名：** | **刘小川** |
| **申请学位专业：** | **信息安全** |
| **申请学位类别：** | **工学学士** |
| **指导教师姓名（职称）：** | **赵文（副教授）** |
| **论文提交日期：** | **2020年06月15日** |

**基于机器学习的恶意代码检测软件的开发**

**摘 要**

近几十年来人们的物质生活被极大地改善同时伴随着技术的不断进步，已经迎来了属于大数据及云计算的时代，企业和个人开始逐渐重视起网站以及个人数据的安全。Webshell即网站后门是一种可执行任意命令的工具它通常以多种形式存在。攻击者成功攻击网站服务后通常会留下Webshell进行持续权限控制，以达到敏感信息获取，及深入内网渗透等目的。所以阻止攻击者攻击并且准确识别攻击者留下的Webshell对企业安全极为重要。

大多数安全产品都基于特征规则匹配（将文件和预先写好的规则代码集合进行匹配），但特征规则匹配对于混淆的Webshell以及脚本语言新特性的检测正确率并不是特别高。

本文提出一种新型的PHP opcode模型，相比较于传统特征规则匹配有更高的抗干扰性，并结合机器学习通过大量数据的训练最终实现检测Webshell的功能。本系统同时采用Flask框架设计出一个便于用户操作使用的恶意代码检测系统，结合用户登录注册，留言建议，后台管理等功能优化用户体验。

最后结果表明系统非常方便用户使用，同时提供批量检测功能，检测结果稳定性好且正确率高。

**关键词：**安全；Webshell检测；机器学习； Python

**Development of Malicious Code Detection Software Based on Machine Learning**

**Abstract**

In recent decades, people's material life has been greatly improved and accompanied by continuous technological advancement. We welcome the era of big data and cloud computing. Enterprises and individuals have begun to pay more attention to the security of websites and personal data. Webshell can be simply understood as a backdoor of a website, and is generally a tool that can execute arbitrary commands in the form of various web page files. After an attacker successfully attacks a website service, the Webshell is usually left for continuous permission control to achieve the purpose of obtaining sensitive information and penetrating into the intranet. Therefore, organizing attackers to attack and detect the Webshell left by the attackers is extremely important for enterprise security. Most security products are based on feature rule matching (matching files with pre-written rule code sets), but feature rule matching is confusing. The detection accuracy of the new features of Webshell and scripting languages is not particularly high.

This paper proposes a new type of PHP opcode model, which is more resistant to interference than traditional feature rule matching. Through the training of large amounts of data with machine learning, the purpose of detecting Webshell is finally achieved. At the same time, the system uses the Flask framework to design a malicious code detection system that is convenient for users to use. It combines user login and registration, message suggestions, background management and other functions to optimize the user experience.

The final result shows that the system is very convenient for users to use, and it also provides batch detection function. The detection result has good stability and high accuracy.

**Key words:** security; webshell detection; machine learning; Python

**目 录**

论文总页数：33页

[1 引言 1](#_Toc40725369)

[1.1 课题背景 1](#_Toc40725370)

[1.1.1 Webshell 1](#_Toc40725371)

[1.1.2 Webshell分类 1](#_Toc40725372)

[1.1.3 Webshell绕过一般检测方法 2](#_Toc40725373)

[1.1.4 机器学习 4](#_Toc40725374)

[1.2 国内外研究现状 5](#_Toc40725375)

[1.2.1 Webshell检测方案 5](#_Toc40725376)

[1.2.2 机器学习用于Webshell检测 6](#_Toc40725377)

[1.3 本课题研究的意义 6](#_Toc40725378)

[1.4 本课题的研究内容 6](#_Toc40725379)

[2 相关技术概述 7](#_Toc40725380)

[2.1 相关机器学习算法原理简介 7](#_Toc40725381)

[2.1.1 朴素贝叶斯算法 7](#_Toc40725382)

[2.1.2 支持向量机算法 7](#_Toc40725383)

[2.1.3 随机森林算法 8](#_Toc40725384)

[2.2 Python介绍 8](#_Toc40725385)

[2.3 Bootstrap介绍 9](#_Toc40725386)

[3 恶意代码检测项目需求分析 9](#_Toc40725387)

[3.1 运行环境 9](#_Toc40725388)

[3.2 功能需求 9](#_Toc40725389)

[3.2.1 登录注册模块 10](#_Toc40725390)

[3.2.2 留言模块 10](#_Toc40725391)

[3.2.3 API调用 10](#_Toc40725392)

[3.2.4 检测模块 10](#_Toc40725393)

[3.2.5 日志记录模块 11](#_Toc40725394)

[3.2.6 管理模块 11](#_Toc40725395)

[3.3 非功能需求 11](#_Toc40725396)

[3.3.1 性能需求 11](#_Toc40725397)

[3.3.2 可维护性需求 11](#_Toc40725398)

[3.3.3 易用性需求 12](#_Toc40725399)

[4 恶意代码检测项目设计与开发 12](#_Toc40725400)

[4.1 登录注册模块设计与实现 12](#_Toc40725401)

[4.2 留言模块设计与实现 15](#_Toc40725402)

[4.3 检测模块的设计与实现 16](#_Toc40725403)

[4.3.1 样本选择 16](#_Toc40725404)

[4.3.2 特征提取 16](#_Toc40725405)

[4.3.3 算法的选择以及调参优化 18](#_Toc40725406)

[4.3.4 检测模块 22](#_Toc40725407)

[4.4 日志模块的设计与实现 23](#_Toc40725408)

[4.5 API调用的设计与实现 23](#_Toc40725409)

[4.6 管理模块的设计与实现 23](#_Toc40725410)

[5 恶意代码检测软件测试 24](#_Toc40725411)

[结 语 30](#_Toc40725412)

[参考文献 31](#_Toc40725413)

[致 谢 32](#_Toc40725414)

[声 明 33](#_Toc40725415)

1 引言

1.1 课题背景

对人们来说Web在如今这个信息爆炸的时代已然成了不可缺少的工具，孩子们通过网络玩游戏，网上冲浪，交友，大人们则通过互联网浏览各种新鲜消息，欣赏歌曲等。当我们在使用便捷的网络工具的时候一般都会填写一些用户信息，手机号，身份证等，用户信息安全对于企业来说至关重要，甚至是有些企业安身立业之本，同样信息安全对用户来说也十分重要，信息的泄漏会造成个人身份信息被盗用以及陌生短信和电话骚扰等问题，当然每个人都不希望自己的信息遭到泄漏所以我们应该重视信息保护。

Web技术作为一把双刃剑，在人们生活更加便捷的同时，也对企业系统的安全造成了很大影响，Web安全变得越来越重要，Web漏洞如SQL注入和XSS攻击等是一些常见的安全问题，这些攻击不仅发生在企业和个人身上，而且发生在政府组织上。攻击者通过Web缺陷获取到服务器权限后通常会将恶意文件或恶意代码（即Webshell）上传到Web服务器上， Webshell被攻击者用作管理和操作服务器的后门工具。有了Webshell后，攻击者可以进一步获取到企业内部或者用户敏感信息，以及进一步深入进行内网渗透。因此，准确并且有效地判断服务器上的文件是否包含恶意代码对Web服务器安全乃至企业安全至关重要。

* + 1. Webshell

PHP和JSP等多种网页编程语言编写Webshell通常以网页文件的形式出现，在最早的设计中Webshell作为网站管理员的强大远程管理工具，管理员可以通过Webshell远程管理网站。但随即技术不断地发展攻击者也开始逐渐关注起Webshell。一般可通过80端口远程访问Webshell。Webshell攻击流程图如图1-1所示。攻击者一般利用网站服务的弱点（例如SQL注入漏洞和文件上传漏洞）获取到服务器权限然后上传Webshell执行恶意代码。Webshell一般包括命令执行、数据库访问、文件操作、网段扫描、特权提升等功能。

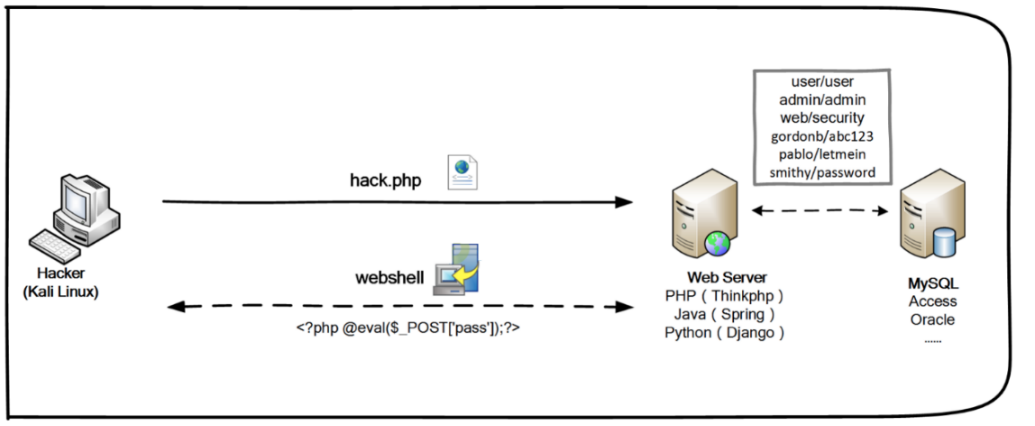


图1-1 Webshell攻击流程

* + 1. Webshell分类

这里以PHP为例Webshell通常依据文件内容和完成的功能被粗略地划分为三类：一句话木马、小马和大马。

（1）一句话木马

由命名可以理解，一句话木马仅包含一句话，此类代码接收攻击者的请求并执行相应的操作，这是最常见的Webshell形式,此类木马使用非常方便，可以单独作为一个文件，还可以嵌入其他正常的文件。

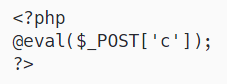


图1-2 一句话木马

在图1-2示例中，攻击者可以访问到Webshell页面，并通过post请求控制c参数进行代码执行来攻击受害者的服务器。例如攻击者可以编写c=phpinfo()来获取当前服务器的PHP的相关信息。通过网页直接操作非常麻烦，通常一句话木马会配合其他三方工具来进行连接并管理。此类代码检测非常容易，一般只需要检测文件里是否包含eval和assert等关键字符即可。

（2）小马

小马用于大马的跳板，通常只包含文件上传功能，攻击者通过小马上传大马，图1-3包含小马的关键代码。

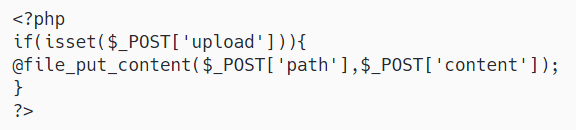


图1-3 小马

（3）大马

大马通常体积很大，1M~2M左右，功能强大且丰富，提供文件操作、端口扫描、数据库操作等多种功能方便攻击者控制服务器，一般不使用工具来连接，直接访问就可以操作。

* + 1. Webshell绕过一般检测方法

（1）拆分

在检测一句话木马的时候通常检查有没有出现eval和assert关键字，那么结合PHP语法，通过先拆分再合并的方法，让脚本文件中不存在会被检测到关键字，就可以绕过检测。如图1-4，这里应用拆分的方法将assert拆分在req变量里，然后使用req变量就相当于使用的assert函数。

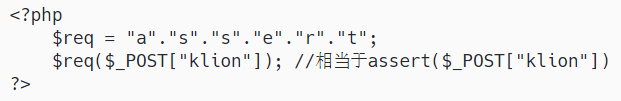


图1-4 拆分绕过检测

（2）混淆

和拆分的思路差不多，只要绕过关键词，那么就不会被检测到，这里结合一些特殊的PHP编码函数比如base64，rot13，ascii码转换等，将恶意代码先通过编码函数转换，运行的时候再转换回来，即可绕过检测同样的原理我们也可以采用一些特殊的方法如图1-5这里使用了位运算中的^（异或符），不使用英文大小写字符，通过一些特殊字符异或的方法，构造出需要的大小写字符，这里的$\_就相当于assert，使用$\_就相当于使用了assert函数，绕过检测，达到了代码执行的目的。

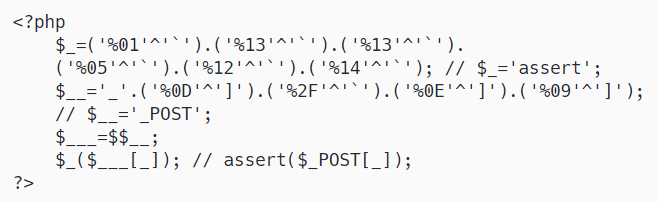


图1-5 编码混淆绕过检测

（3）藏匿于其他文件

当文件大小过大时，有些检测软件可能为了提高效率，直接跳过对文件的检测，那么恶意代码就可以隐藏于这些文件当中，比如藏在PHP文件的注释里以及Web服务的运行日志文件里。然后与PHP的文件包含漏洞相配合就可以使这些藏在其他文件中的恶意代码就可以被执行。

常见的比如图片马，可以使用Windows dos命令行下的copy命令制作，这样的文件和一般的图片文件看起来没有什么区别，使用图片浏览工具也可以正常打开，使用一些特殊的二进制查看工具例如winhex，010editor就可以查看到里面隐藏的恶意代码，如图1-6所示。因为这些恶意代码隐藏在图片当中，文件数据中含有图片头，那么有些检测软件检测的时候会直接忽略，从而绕过检测。

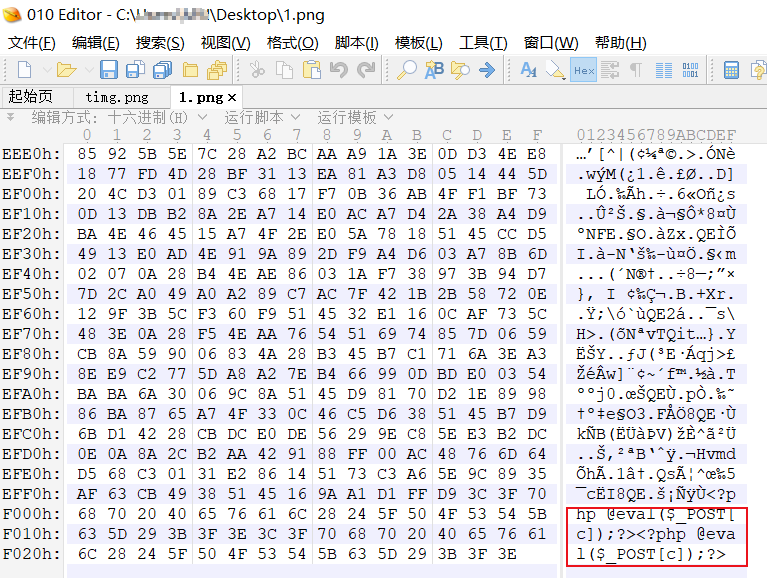


图1-6 图片马

* + 1. 机器学习

关于机器学习最根本的是让计算机使用自己的能力去学习。机器学习使用的算法目的在于从一大块数据中发现其中隐藏的规则，然后利用它来预测新的数据。这些算法与很多统计学知识有关，所以想要更加深入了解机器学习，必然也需要统计学的学习。如今很多领域都选用机器学习来作为技术支持。机器学习主要分为监督学习（Supervised Learning），无监督学习（Unsupervised Learning）和强化学习三大分类。

（1）监督学习

在这种学习方式里已知很多数据（这些数据包含个体的特征），并且还可以清楚地把这些个体划分到不同分类，这些特征和标记的关系就是我们想要的到的即通过含有标记的数据学习数据中所隐含的规律。

回归问题以及分类问题是监督学习的两大方向关于回归问题比如说现在想要投资一块房地产，已知一些地产规格大小，周围环境以及离市区距离还有它们最后的获利，我拟合出一个函数把相应的数据输入到这个函数就可以预估投资这个房地产大概的利润这个过程就是解决回归问题的过程。

接下来关于分类的问题，我现在打开邮箱获取里面的邮件这里面包含垃圾邮件和正常的邮件，现在我分别提取出这些邮件中的特征（词集划分，获取分词），然后我通过机器学习根据之前的方法同样拟合出一个函数，下次我再收到邮件时，就将邮件输入到这个函数然后根据他的输出结果来预测这封邮件是不是垃圾邮件以上过程就是解决分类问题的过程。

回归问题和分类问题是不同的其中的差别在于一般的回归问题是获取一个特定的结果（值等）而分类问题是根据数据为样本划分一个属于哪一类的标记。

（2）无监督学习

无监督学习就是在大量数据中根据数据间本身的关系将他们划分成一些类。关于监督学习在这之前是知道数据属于哪个分类然而无监督学习只知道有这么一些数据并不知道他们分属哪一类。也就是说，无监督学习根据未标记的数据发现其中隐藏的规律。

例如，每个新闻平台（头条、新华社、微博、微信公众号）都有许多新闻项目，这些新闻项目都可以通过无监督学习来收集分类。然后之后有人再发表新闻的时候就可以使用学习的结果来判断他写的新闻该划分到哪一类。

（3）强化学习

这种学习方式和监督学习相同的点在于都需要海量数据去完善训练模型。他们之间的区别在于：（1）两者的学习方式不尽相同（2）两者所需要的数据类型存在差异，监督学习要求大量带有分类结果的数据然而强化学习想要数据结果能回馈模型使模型更精确。

1.2 国内外研究现状

1.2.1 Webshell检测方案

为了检测服务器上Webshell同时保障安全，随着时间的推移常见的Webshell检测方法包括人工检测、静态检测、动态检测、日志分析、统计学检测五种。

（1）人工检测

手工检测是webshell最早的一种检测方法，它要求web管理员具有更高的能力，能够对网站新增的后缀为php和jsp等的脚本文件进行排查，通常需要花费大量的时间，一般的方法是查找一些PHP的危险操作函数比如eval以及exec等。

（2）静态检测

在人工检测之后就出现了根据静态特征匹配的恶意代码检测技术，这种技术可快速检测快速定位也是当前大多数检测软件所采用的方法，静态检测和人工检测其实差不多，只不过使用了机器代替了人工检测，需要编写大量的规则库，首先这对规则编写人员要求还是比较高其次基本检测不到新出现的Webshell特征。

（3）动态检测

Webshell运行后类似于C/S服务模式那么可以尝试从HTTP流量方面进行检测。Webshell在运行时会使用系统命令或者其他的一些异常操作，根据脚本文件的具体操作来判断其是否安全。这种方式可以检测到一些比较新的或使用了新特性的恶意脚本但是现在很多Webshell会对流量进行加密传输这无疑增加了动态检测的难度。对于特殊功能的Webshll的检测难度也比较大困难。而且因为是基于流量的被动检测，恶意代码已经执行，有可能信息已经被窃取，这时候才检测到，为时已晚。如果一个网站一天的流量非常的大，那么流量特征检测的效率就会非常的低，甚至还会影响到正常业务逻辑的使用。

（4）日志分析

攻击者使用完Web后门过后通常在系统里查询不到日志，但是在网站的Web服务器的访问日志一般都会留下访问日志。通过大量访问日志的分析构建出HTTP异常请求访问模型。比如，一个地址基本通常接收GET类型的请求，有一天却突然出现了其他类型的请求，此时我们记录访问来源的IP地址以及访问时间等并以此为依据生成访问者肖像。这种日志分析的方式在大量的请求的支持下检测效果较好，，但也存在误报的情况，且对于新发展的业务环境的检测效果并不是很好。类似于动态检测的缺点，从日志方向分析也是一种被动的方法，并且已经造成了危害。对于大量的日志，检测效率也有可能下降。

（5）统计学检测

很多检测系统现在比较喜欢采用统计学的方法，提取文件中的特征代码，然后，我们使用统计信息熵、最长单词、符合索引、特征、压缩等方法查找可能混淆或编码的恶意代码，这种方法在识别模糊代码或被混淆的代码时表现结果较好，但常常误报率也很高。

其余的文件属性，例如文件是什么时候修改创建等，这些并不针对文件检测本身，而是作为一些辅助信息。

1.2.2 机器学习用于Webshell检测

随着大数据和人工智能时代的到来机器学习逐渐被用于安全业务关于Webshell检测大致结合上文所述检测方法，结合文本特征、流量特征、日志访问特征等然后采用机器学习的方法，通过对大量样本的学习的和训练甚至结合深度学习以及神经网络等完成对Webshell的检测。

1.3 本课题研究的意义

仅对于基于文本特征的机器学习检测Webshll来说，一般是对文件文件内容构建词袋模型来获取相关特征，但由于脚本文件中注释等代码块的影响，对文件的检测正确率并不是很高，而且对于新版本脚本语言的特征的适应度也并不是很好。本课题提出的PHP opcode模型能很好地克服传统基于文本特征的机器学习检测Webshll的缺点，提高检测的正确率和效率，同时基于研究结果还会设置出一个完整的方便用户操作的Webshell检测平台。

1.4 本课题的研究内容

首先研究了Webshell的分类和常见的检测方法以及机器学习用户Webshell检测的常见方式，在了解基本优缺点的基础上，确定了主要实现目标即基于PHP opcode模型的Webshell检测的研究与实现。

通过资料收集，文献阅读，大致需要实现以下功能：

（1）检测模块

此模块通过构建PHP opcode模型然后提取其特征之后通过机器学习的方法对大量数据经行学习训练，最后构建出可用于直接预测是否包含恶意代码的模型。

（2）便于用户操作的检测平台

该平台由用户登录注册、消息、后台管理等模块组成，主要用于将功能一所构建的模型的便捷使用，同时方便用户操作以及交流讨论。

2 相关技术概述

2.1 相关机器学习算法原理简介

2.1.1 朴素贝叶斯算法

关于朴素贝叶斯算法，其核心是在贝叶斯公式之上的，式2-1，将B理解为类别，将A理解为特征，那么通过贝叶斯公式就可以得到P(特征|类别)到P(类别|特征)的转换即计算一个特定特征属于每一个类别的概率然后对比这些概率，概率最大那么就可以把具有这个特征的事物分到那一类中。

（式2-1）

朴素贝叶斯算法是建立在单独特征是彼此没有影响的基础上的，那么就可以得到式2-2，基于大数定理，即可粗略地将不同特征的事物分类。

（式2-2）

2.1.2 支持向量机算法

支持向量机算法的目的是要确定一个超平面来划分标记，通过距离等的计算，它使用最优超平面需使间隔最大化，如图2-1，如果在一维上不可分，那么可以通过引入核函数的方式将数据提升到更高维然后再尝试去划分。

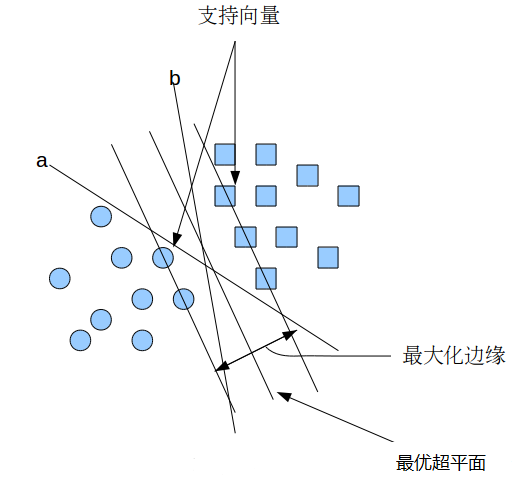


图2-1 支持向量机算法

2.1.3 随机森林算法

随机森林算法基于决策树算法这种算法首先随机抽样训练决策树，随机选取属性分裂，然后重复上面步骤知道不能再分类，最后建立大量决策树生成森林。

算法在预测的时候输入新的样本，让每个决策树对随机森林中的样本进行分类然后将最多的分类结果作为最终结果。

2.2 Python介绍

Python是多范式编程语言支持面向对象的编程并使编程结构化以及继承，重载等功能，有利于提高代码的复用性，内置了很多标准模块这样很大程度降低了开发者工作的重复性，支持字典，集合和生成器。和Java，C++等语言等语言对比开发者能够选择更简洁的程序来表现自己的独特的思想且代码缩进风格统一，Python试图让小型程序或大型程序的结构都可以简洁易读受到广大开发者的喜爱，如图2-2所示，Python在2020年4月PYPL发布的编程语言排行榜占据榜首，超第二名Java语言12%。

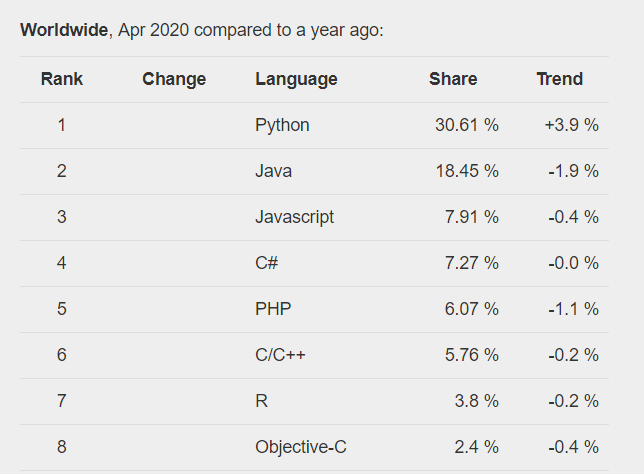


图2-2 PYPL排行榜

虽然Python被粗略地分为脚本语言但是很多大型的开发厂商比如Google广泛地使用Python，是一种高端动态编程语言，它常用于网站应用开发、可视化程序开发、操作系统以及科学计算。

Python提供了很多内置的非常有用的库同时有很多开发者也为Python语言添砖加瓦，提供了很多可以完成更复杂功能的第三方库，这里根据项目的使用情况，介绍几个第三方库：

（1）Flask

Flask是一款小巧的方便快捷的应用框架是在Jinja模版和WSGI的基础开发实现的，Flask有蓬勃发展的社区和强大的拓展，与此同时它方便入手，结构简单，收到广大开发者的喜爱。

（2）SQLALchemy

SQLALchemy 在Python中很有名其ORM操作将数据库中的关系转化成类，操作类中的数据对应操作数据库中列的数据也会被修改，数据库表被抽象为对象这样的操作让程序也变得更加简洁易读。封装了很多数据库引擎，只需要配置数据库连接符，就可以方便地控制数据库数据同时能够有效避免SQL注入，增强了安全性。

（3）Sklearn

Sklearn是Python开发的用于机器学习的建立在Numpy，Scipy，Pandas之上的专门工具。它提供了很多关于数据处理的工具同时Sklearn内置了大量机器学习算法的实现，可以通过简单的调用实现一些简单的机器学习算法比如SVM及KNN等。

2.3 Bootstrap介绍

Bootstrap是基于最新Web网页技术开发的方便快捷的前端开发框架，支持移动设备自动更改显示效果同时提供大量功能强大的内置组件并且方便定制为开发者提供了简洁统一的解决方案。

3 恶意代码检测项目需求分析

3.1 运行环境

系统版本要求：Windows7及以上

运行环境：Python3.6+

开发工具：Jetbrain Pycharm 2019.3

3.2 功能需求

本程序主要分登录注册模块、留言模块、API调用、检测模块、日志记录模块、管理模块几部分组成如图3-1所示。

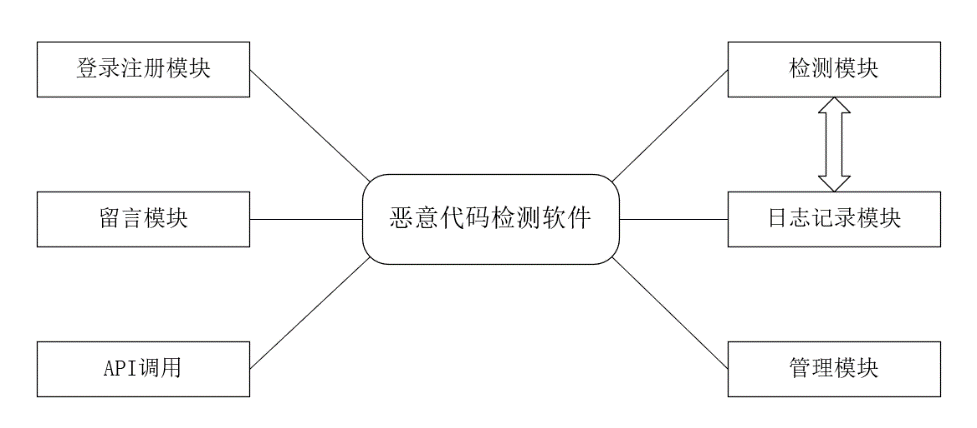


图3-1 系统总体模块设计

3.2.1 登录注册模块

这个模块主要功能是完成用户帐号的登录注册等功能操作，如图3-2所示，各个功能的说明如下。

（1）用户登录：在输入并确认系统所需要的各种数据后，点击登录按钮，经过后台验证成功，用户的浏览器地址会跳转到首页界面，否则会进行提示。

（2）用户注册：输入页面提示的所需信息并确认密码无误后，点击确认，之后系统会经行格式验证，成功之后会在数据库添加用户信息，如遇邮箱已注册，错误等信息会在页面有提示。

（3）忘记密码：如果不小心遗忘了密码，那么按照页面提示操作之后，用户就可以在自己邮箱收到链接，通过访问就可以对密码进行重置操作，邮箱未注册等错误情况会在页面提示。

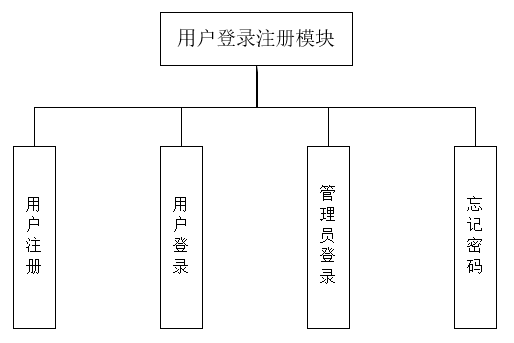


图3-2 登录注册模块

3.2.2 留言模块

在首页，用户可以对系统设计，检测结果等提出建议，进行交流，留言之后会在主页显示用户名，留言内容和留言时间等信息。

3.2.3 API调用

为了避免用户登录注册以及方便应用编程等的需求，提供了API接口调用，仅需要发送一个POST请求，带上检测的算法信息，用户的secret\_key（防止被恶意调用占用服务器资源，可以在用户资料页面查看到自己的secret\_key参数值）以及需要检测内容的base64编码值即可以方便快捷地进行检测，返回json格式的检测结果，方便后续编程处理。

3.2.4 检测模块

可通过输入文件内容以及上传文件两种方式进行检测，同时支持多文件上传进行批量检测且要求单个文件大小不大于5M，检测完成后检测结果会在页面显示。

3.2.5 日志记录模块

检测的时候会自动记录检测时间，保存的文件名以及检测标记等信息到数据库，方便将用户检查的结果加入训练集进行进一步训练，同时在用户资料界面可以查看到检测的记录，可以下载检测的文件，同时还支持更正检测的结果，方便训练结果的进一步优化。

3.2.6 管理模块

管理模块主要完成对网站页面管理其基本功能如图3-3所示，评论管理，检测模型管理等内容，具体功能需求如下：

（1）网站页面管理：可对更新日志，API界面和Wiki界面进行管理显示，更新日志主要展示网站的一些更新进度及改进等，API显示API的调用说明，方便用户使用，Wiki页面主要对网站使用进行介绍。

（2）评论管理：可查看评论详情，评论人，邮箱，评论时间等信息，可删除相应数据。

（3）模型管理：模型管理分为两大块功能，第一块，对检测使用模型的管理，管理员可以在管理界面选择每种算法使用的模型，方便应用再训练的结果。第二块，再训练生成新模型，管理员可以选择使用的白样本和黑样本，同时再训练时会提取用户在前台上传的文件，也加入样本中进行再训练学习。

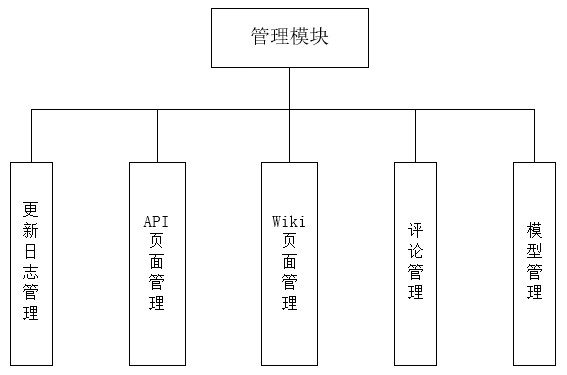


图3-3 管理模块

3.3 非功能需求

3.3.1 性能需求

基本页面响应时间不超过3秒，网页链接及按钮点击能很快返回。

3.3.2 可维护性需求

编写代码过程中遵守相关代码规范，代码简洁明了，各模块有些部分可互用。

3.3.3 易用性需求

用户与系统交互过程中做到页面干净整洁，简单易懂，且对用户不当操作有相关错误提示。

4 恶意代码检测项目设计与开发

4.1 登录注册模块设计与实现

（1）登录功能

访问者在输入邮箱和密码之后选择登录，通过检测之后，后台会通过hash值的方式验证用户密码验证成功之后跳转到平台首页，相应的凭证会被存储到session中，流程图如图4-1所示，Flask中可以使用current\_user获取到当前用户信息，同时如果一个已经登录的用户访问登录页面时系统会进行判断然后自动跳转到平台首页不再进行登录操作。

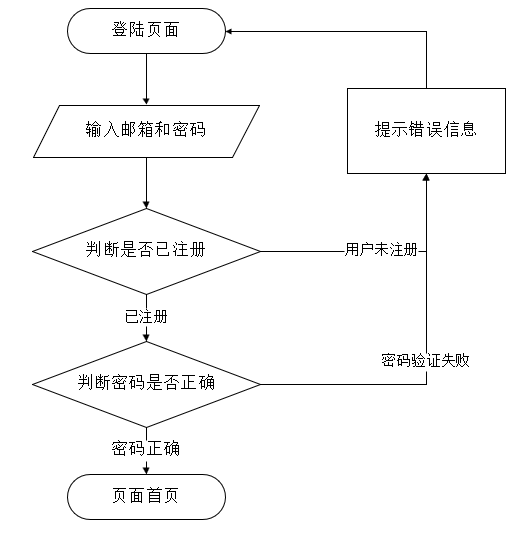


图4-1 登录流程

核心代码实现：

@auth\_bp.route('/my\_own\_login', methods=['GET', 'POST'])

def login():

if current\_user.is\_authenticated:

return redirect(url\_for('app.index'))

login\_form = LoginForm()

if login\_form.validate\_on\_submit():

email\_addr = login\_form.email.data

password = login\_form.password.data

user = User.query.filter\_by(email=email\_addr).first()

if user.validate\_password(password):

login\_user(user)

return redirect(url\_for('app.index'))

else:

flash('password error', 'password\_error')

return render\_template('auth/login.html', login\_form=login\_form)

validate\_password函数将用户输入的密码转换成hash值并与要登录的用户进行匹配，具体会在用户注册功能讲到，login\_user是Flask login模块提供的函数会将用户登录状态注册到session中。

（2）注册功能

进入注册界面用户会被要求帐号相关的密码等信息，需要输入两次密码进行确认，如果两次密码不相同则会进行提示。且如果邮箱账号之前已经注册过也会提示相关信息，注册流程如图4-2所示。

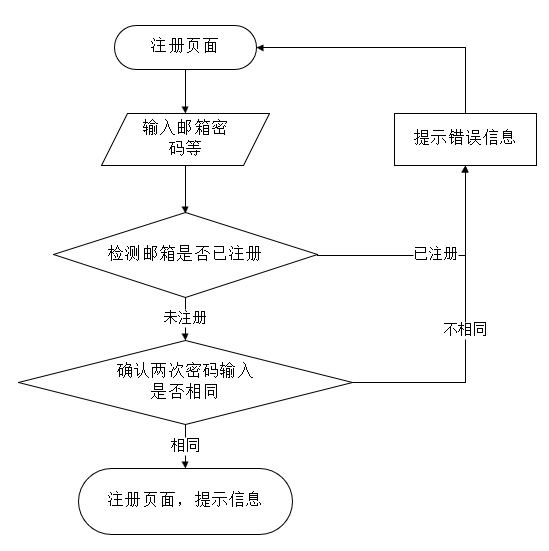


图4-2 注册流程

核心代码实现：

@auth\_bp.route('/register', methods=['GET', 'POST'])

def the\_my\_register():

if current\_user.is\_authenticated:

return redirect(url\_for('app.index'))

register\_form = RegisterForm()

if register\_form.validate\_on\_submit():

……

user = User(email=email\_addr, username=username, secret\_key=generate\_secret\_key(11))

user.set\_password(password)

……

flash('register success', 'register success')

return redirect(url\_for('auth.register'))

return render\_template('auth/register.html', register\_form=register\_form)

因为使用了ORM技术我们实例化一个User对象，User对象即对应在user表中添加一条记录user表的结构如表4-1所示。set\_password和validate\_password函数是User类中编写的函数，主要是使用werkzeug的安全模块实现。

表4-1 user表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | user | | | | |
| 字段名 | 字段类型 | 描述 | 是否为主键 | 是否可以为空 | 默认值 |
| \_id | Onject ID | id号 | 是 | 否 | 无 |
| email | String | 邮箱地址 | 否 | 否 | 无 |
| username | String | 用户名 | 否 | 否 | 无 |
| passwd\_hash | String | 密码hash | 否 | 否 | 无 |
| secret\_key | String | api调用密匙 | 否 | 否 | 无 |
| is\_admin | Boolean | 判断管理员 | 否 | 否 | false |

（3）找回密码

用户输入注册时填写的邮箱号然后点击相应控制按钮，会收到修改密码的链接地址，访问地址即可实现对密码修改的操作。如果邮箱地址未注册，会在页面进行提示。

核心代码实现如图4-3

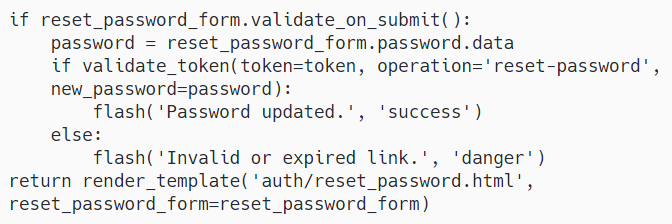


图4-3 找回密码实现

token由Python Serializer模块生成，里面信息包含用户信息，执行操作，过期时间等，validate\_token会将token内容反序列化出来，检测用户信息，是否过期等。

4.2 留言模块设计与实现

留言内容被抽象成一个Comments对象，包含用户id，用户名，邮件地址，留言内容，留言时间等，用户点击留言时会实例化一个Comments对象，存储在数据库中，然后会在首页被渲染显示出来。

核心代码实现：

@app\_bp.route('/', methods=['GET', 'POST'])

@login\_required

def index():

……

comments = Comments.query.all()

changelog = ChangeLog.query.order\_by(ChangeLog.id.desc()).first()

……

return render\_template('app/index.html', comments=comments, ……)

#渲染代码

<ul class="list-group">

{% for comment in comments %}

<li class="……">

<div class="……">

<small …… data-placement="top">

{{ comment.time }}

</small>

<h5 class="m-b-1">

{{ comment.username }}({{ comment.email }})

</h5>

</div>

<p class="m-b-1" ……>{{ comment.content }}</p>

</li>

{% endfor %}

</ul>

comments获取Comments的所有对象，然后再传递到index模版中，通过模版渲染愈发，将留言信息显示在页面上。

4.3 检测模块的设计与实现

4.3.1 样本选择

通过爬虫以及网络搜索等多种途径，搜集到关于PHP的相关项目，白名单选用laravel、phpmyadmin、typecho、wordpress、yii2、PHPyun、thinkphp等PHP开源框架及项目，黑名单选用github，码云以及网络途径整理的PHP Webshell样本库。同时为了增加样本数量以及多样性，使用php-venom项目（github上一个用户生成混淆的免杀马的项目，蚁剑就使用这个项目），生成了1000个免杀的PHP混淆马，如图4-4所示。最终获取到58354个样本，白样本23568个，黑样本34786个。

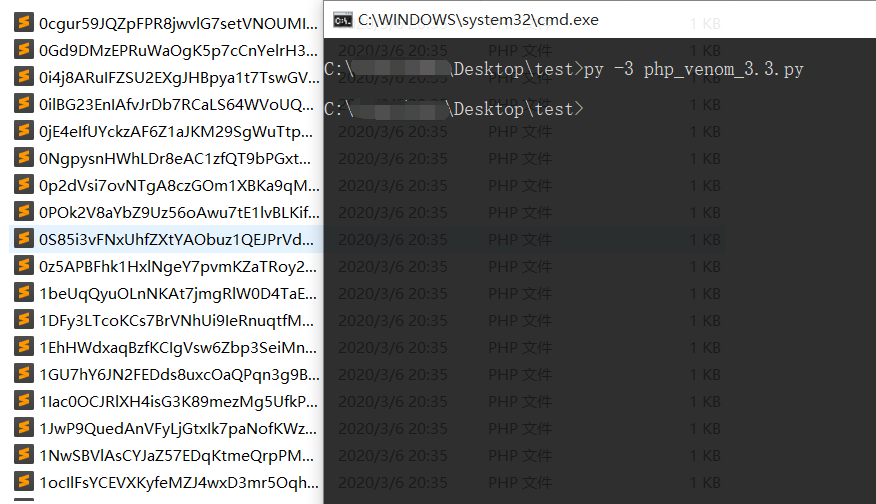


图4-4 生成1000个免杀马

4.3.2 特征提取

首先只使用词集划分的方式对所得数据训练学习，选用朴素贝叶斯算法，发现训练的结果的正确率很低，大概只在80%左右，且对新样本的检测效果误报率也非常的高。

（1）PHP opcode

PHP opcode如图4-5类似于Java的byte code，是介于汇编语言与高级语言之间的形式，PHP规定了每个文件的opcode只能属于哪些，这很好地避免了注释内容影响同时也大大提高了特征提取的效率于是选用了采取PHP opcode模型的方法，先提取每个PHP文件对应的opcode然后再进行词集划分。

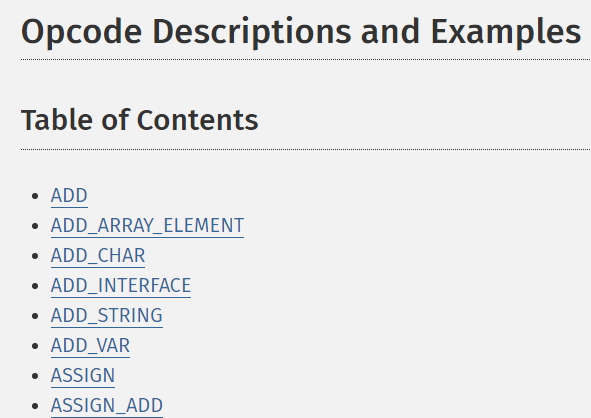


图4-5 opcode

首先安装PHP dvld拓展，并在php.ini配置文件里开启拓展，然后就可以通过命令行的方式提取每一个文件对应的opcode，如图4-6。

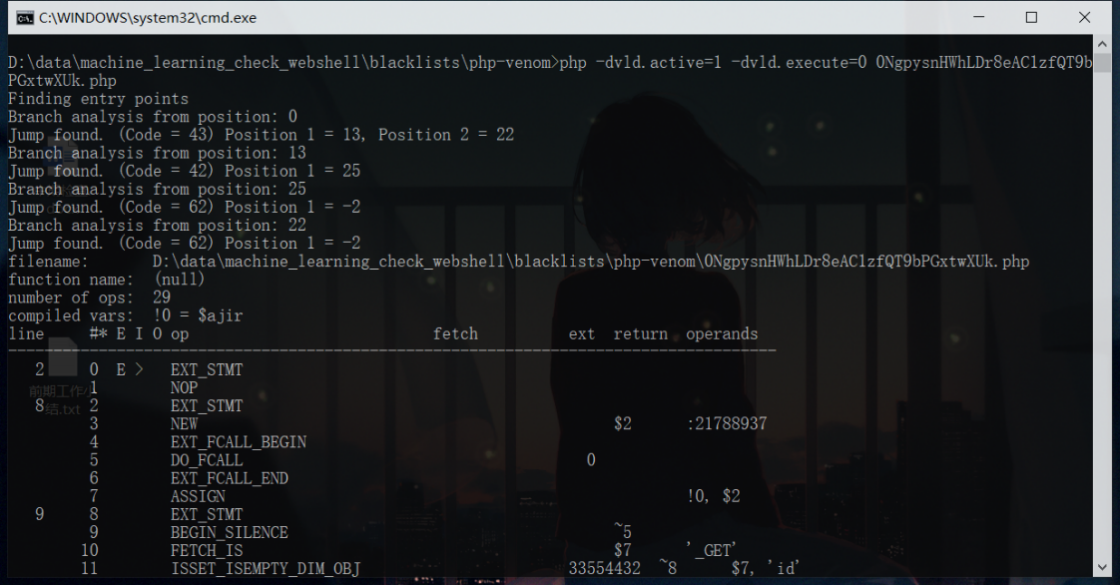


图4-6 提取PHP opcode

当然挨个通过命令行的方式效率太低，于是使用Python的子程序模块获取命令行输出并结合正则表达式匹配的方式忽略其他符号影响。

核心代码实现：

output = subprocess.check\_output(['php.exe', '-dvld.active=1', '-dvld.execute=0', phpfilename], stderr=subprocess.STDOUT)

tokens = re.findall(r'\s\b([A-Z\_]{2,})\b\s', output.decode())

t = " ".join(tokens)

return t

因为PHP 5.4以下会导致兼容性问题，导致提取opcode失败，如图4-7，所以对于这类情况返回空列表。

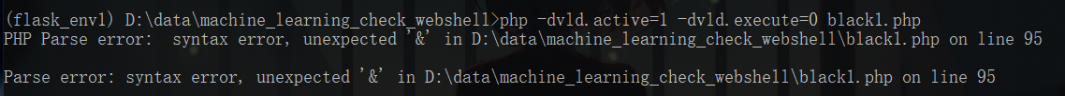


图4-7 PHP 5.4以下兼容问题

（2）词袋和TF-IDF模型

词袋模型是用于自然语言处理（NLP）的特征提取模型，根据向量维数来对文档进行划分，然后根据划分的字典以及文档内容生成词集的特征向量。比如“你吃饭了吗”用2-gram划分，则划分成的字典为“你吃”，“吃饭”，“饭了”，“了吗”。

词袋模型仅统计了单词的频率然而这种统计的方式并不能反映出其重要性，比如说一些语气词和连接词，它们并不表现文本内容，但有可能在文本中出现很多次。于是TF-IDF在词袋模型的基础上进行进一步处理，根据单词的逆文档频率来修改单词特征词，选出识别能力较弱的单词，同时保留重要单词，这种方式获取到的特征矩阵每个特征的差距不会像词袋模型的那么大，TF-IDF的计算如式4-1所示。

 （式4-1）

TF词频对应词在文章中出现总次数和文章总次数的比例其中IDF的计算如式4-2所示。

 （式4-2）

将每个PHP文件对应的opcode通过TD-IDF模型处理，获取到特征向量矩阵，同时对特征向量进行标记。

核心代码实现：

# 打标

X = blacklists + whitelists

y = [1] \* blacklists\_len + [0] \* whitelists\_len

tfidf\_vec = TfidfVectorizer(decode\_error='ignore', token\_pattern=r'\b\w+\b', ngram\_range=(2, 3))

tfidf\_matrix = tfidf\_vec.fit\_transform(X)

4.3.3 算法的选择以及调参优化

在学习了各种分类算法以及实际使用不同算法训练比较正确率等多种因素的基础上，最终选择了朴素贝叶斯、支持向量机、随机森林三种算法用于检测。比如KNN算法它是基于k-近邻有监督的分类的算法但是因为Webshell分类是二分类问题它并不太适用，在实际的测试训练中，也发现KNN算法相比于其他三种算法正确率较低。

对获取到数据进行训练学习然后使用十则交叉验证对朴素贝叶斯算法进行评估，首次正确率为88%，如图4-8。

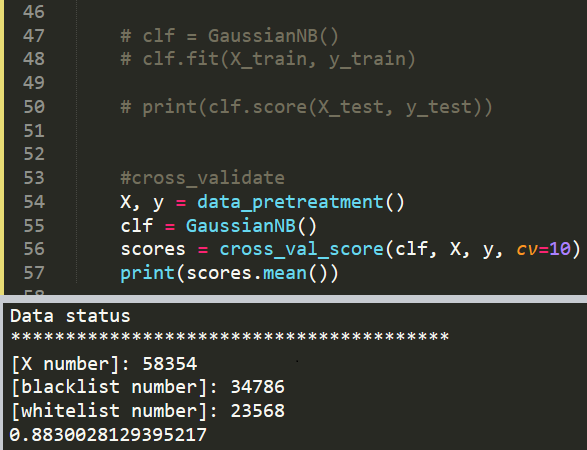


图4-8 朴素贝叶斯算法首次88正确率

因为TF-IDF默认是使用1-gram处理，将处理模型更改为2-gram和3-gram后朴素贝叶斯算法正确率上升到91%，如图4-9。

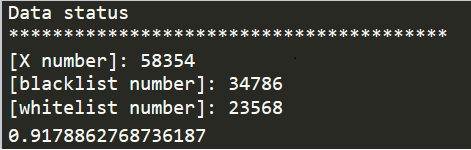


图4-9 调整gram模型后正确率上升

使用SVM算法首次正确率51%左右，如图4-10。

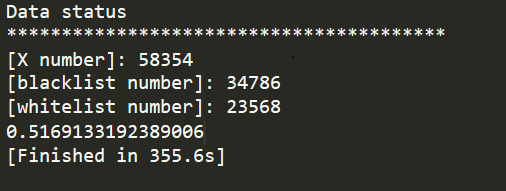


图4-10 SVM首次51正确率

因为SVM默认使用的是rbf高斯核，在更换为线性核后正确率升高到95%，如图4-11。

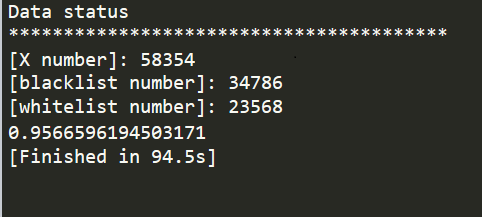


图4-11 更换线性核正确率提升

gamma和C，SVM的两个参数会调整训练结果的正确率，这里换用了线性核，调整gamma就不起作用了，C是惩罚系数。为了避免C参数过高造成过拟合的问题这里对C参数进行调参处理，如图4-12所示，可以看到C 在6.32368421e-09 至9.48052632e-08 之间效果最好。

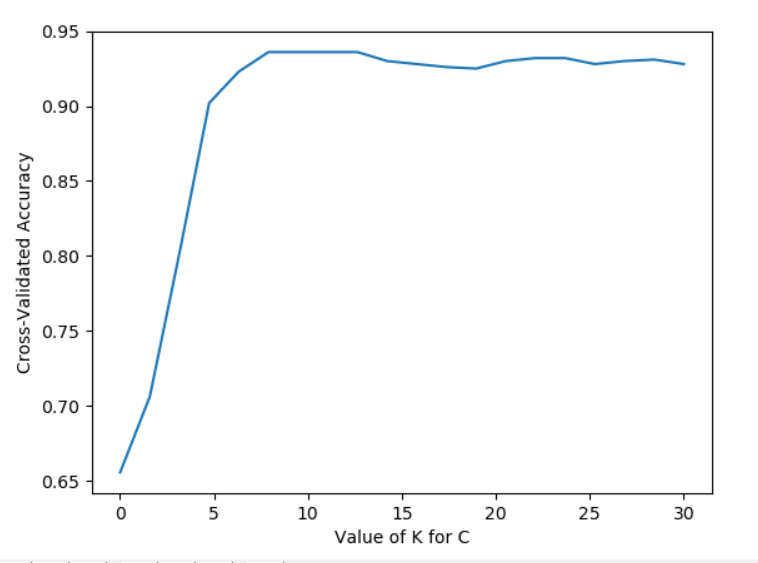


图4-12 C变化得分

调参核心代码：

CList = np.logspace(-10,1,20)

for C in CList:

clf = SVC(kernel='linear', C=C)

scores = cross\_val\_score(clf, X, y, cv=10, scoring='accuracy')

score\_C.append(scores.mean())

print('score: {}'.format(scores.mean()))

随机森林算法未调参时首次十则交叉验证争取率94%，如图4-13。

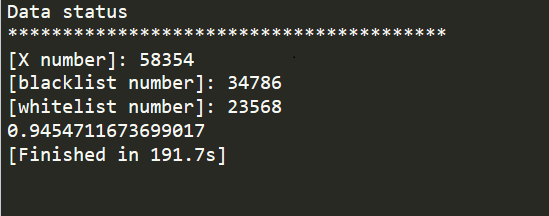


图4-13 随机森林算法首次94正确率

决策树生成的数量会对随机森林的结果产生影响这里对树的数目进行调整，如图4-14所示，可以看到大概大于60效果都比较好。

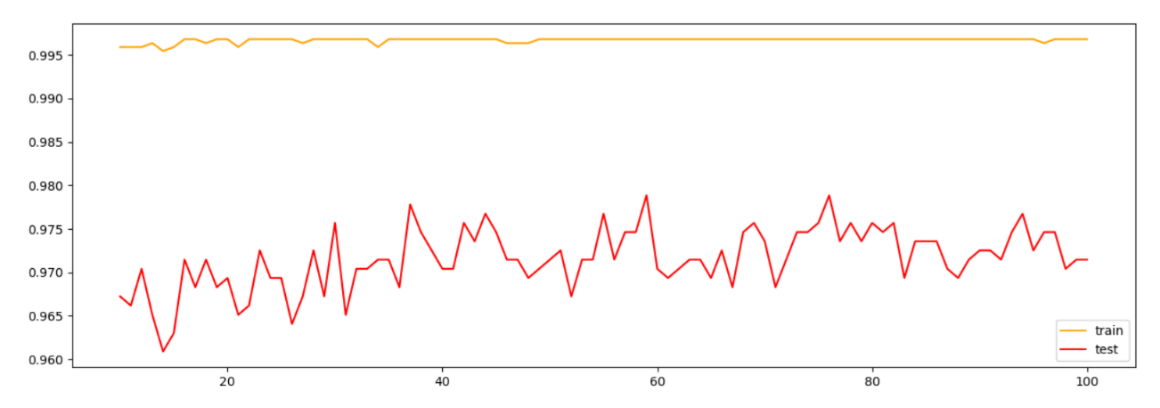


图4-14 随机森林树的数目

调参核心代码实现：

for i in range(10,101):

rfc = RandomForestClassifier(n\_estimators=i, n\_jobs=4).fit(X\_train,y\_train)

train\_score.append(rfc.score(X\_train,y\_train))

test\_score.append(rfc.score(X\_test,y\_test))

plt.figure(figsize=(20, 5))

plt.plot(range(10,101),train\_score,color='orange',label='train')

plt.plot(range(10,101),test\_score,color='red',label='test')

plt.legend(loc='lower right')

plt.show()

调参完成后最终朴素贝叶斯94%正确率、SVM算法97正确率、随机森林算法98正确率，如图4-15所示。

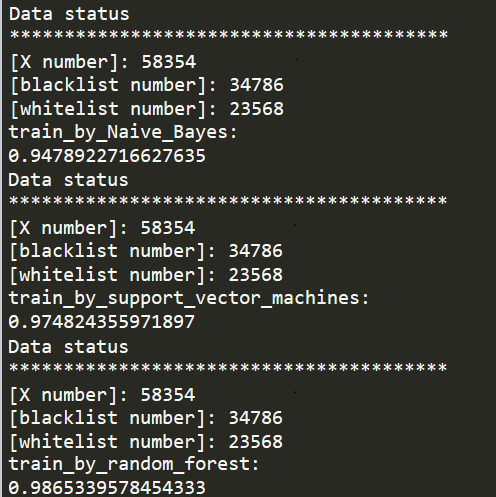


图4-15 最终正确率

4.3.4 检测模块

将训练好之后的模型保存到pikle文件，然后在使用的时候从pikle文件中提取出来，然后通过之前获取特征的方法获取需要检测文件的特征，再使用模型预测即可。这里提供了命令行检测工具，如图4-16，提供单文件，目录检测，输出结果为txt，pdf，csv格式等功能。

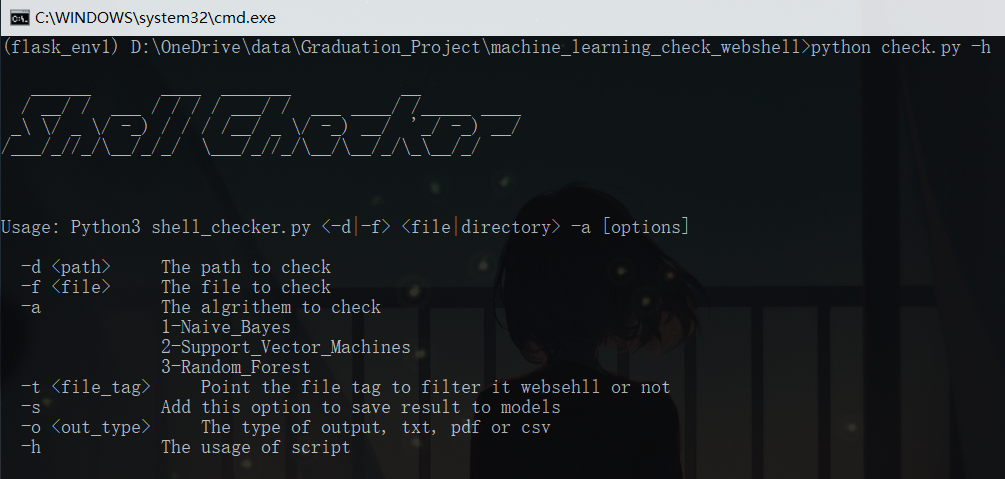


图4-16 命令行检测工具

用户输入文件内容和上传文件两种方式核心都是首先将内容写入到文件中，然后获取文件路径，调用检测模块检测。

核心代码实现：

# 将php\_code写入文件

filename = current\_user.secret\_key + '\_' + random\_filename()

with open(os.path.join(current\_app.config['THE\_PATH\_TO\_UPLOAD'], filename), 'wb') as f:

f.write(php\_code.encode())

# 检测

saved\_file = os.path.join(current\_app.config[' THE\_PATH\_TO\_UPLOAD '], filename)

webshell\_result = check\_webshell(saved\_file, algorithm1)

log\_check(filename, webshell\_result)

random\_filename函数生成随机文件名，采用uuid实现随机，文件后缀名为.sample，避免文件直接执行，check\_webshell会根据algorithm参数自动选择算法模型进行检测并反馈检测结果。

多文件批量检测使用了dropzone组件完成前端展示，后端编写上传逻辑后，将文件逐个存储然后再调用检测模块进行检测。

4.4 日志模块的设计与实现

运用ORM的方法，添加了一个User\_File类，用于存储用户的检测记录，包含文件名，检测时间，检测结果等信息，每次用户通过页面检测时都会自动生成一个对应的User\_File实例，存储在数据库中，用户可以通过个人资料界面查看相关记录，并且可以更改检测结果的标记。

4.5 API调用的设计与实现

API调用也是利用检测模块对用户请求内容进行检测，只不过不是返回页面，而是反馈json字符串。

核心代码实现：

if webshell\_result == [0]:

statue = False

result = 'not webshell'

if webshell\_result == [1]:

statue = True

result = 'webshell detected'

return Response(json.dumps({'statue': statue, 'result': result}), mimetype='application/json')

4.6 管理模块的设计与实现

更新日志，API页面，Wiki界面管理均使用Ckeditor作前端展示和富文本编辑，后台完成修改存储和文件上传等功能，在数据库中显示的内容会以网页文本的形式存储，显示的时候再从数据库读取就可以。

评论删除也是使用ORM类，传入评论id值即可删除数据库中相关数据。

评论删除核心代码实现：

@admin\_bp.route('/delete\_comment/<int:comment\_id>', methods=['POST'])

def the\_my\_delete\_comment(the\_my\_comment\_id):

……

comment = Comments.query.get\_or\_404(comment\_id)

db.session.delete(comment)

db.session.commit()

flash('Delete Success', 'Delete\_Success')

return redirect(url\_for('.comments'))

模型再训练功能会提取管理员选择的白名单样本和黑名单样本具体内容，并且结合日志记录模块记录的文件和响应标记，混合在一起进行再训练。

模型选择功能主要原理是在Flask设置中添加相关配置，管理员修改时使用config.update即可对配置文件中的配置进行修改，检测模块在检测的时候也是读取配置文件中的配置来选择检测模型。

核心代码实现：

if modelform.validate\_on\_submit():

Naive\_Bayes = modelform.Naive\_Bayes.data

Support\_Vector\_Machines = modelform.Support\_Vector\_Machines.data

Random\_Forest = modelform.Random\_Forest.data

current\_app.config.update(

NAIVE\_BAYES\_MODEL=Naive\_Bayes,

SVM\_MODEL=Support\_Vector\_Machines,

RANDOM\_FOREST\_MODEL=Random\_Forest

)

5 恶意代码检测软件测试

（1）登录注册页面，包含邮箱和密码输入框，可进入注册和忘记密码界面，如图5-1



图5-1 登录注册页面

（2）忘记密码页面输入邮箱地址后然后点击发送邮件，收到重置密码邮件，如图5-所示

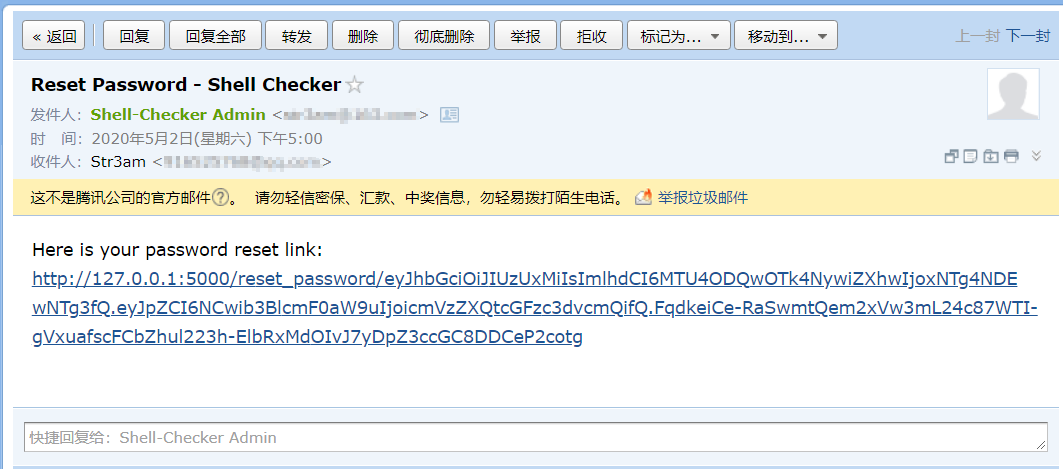


图5-2 重置密码邮件

（2）选取一个混淆马，上传文件方式，选用朴素贝叶斯算法检测，提示检测到Webshell，如图5-3

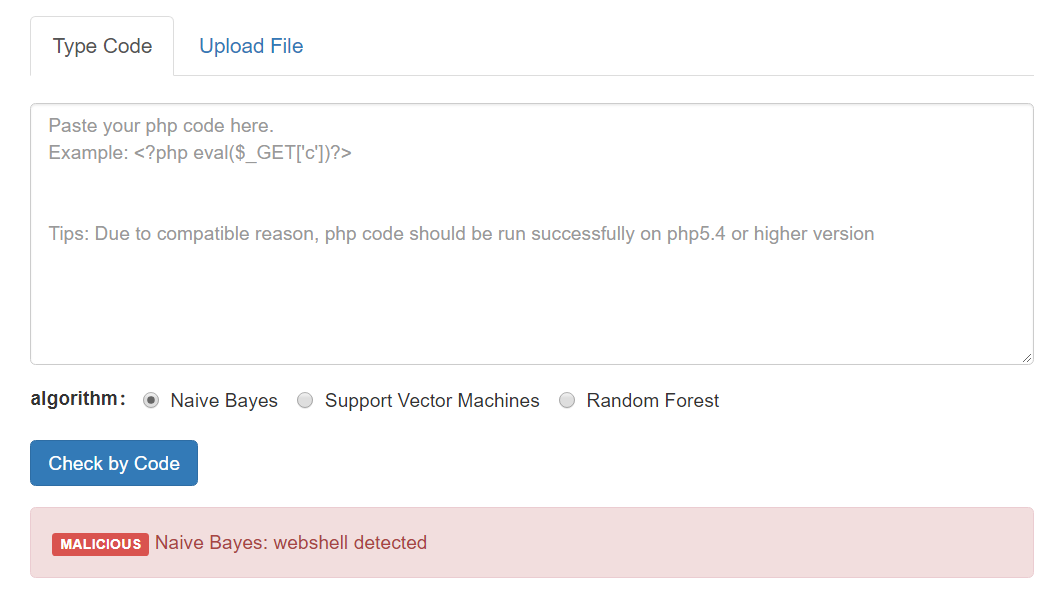


图5-3 Webshell单文件测试

（3）选取一个正常的PHP文件，输入文件代码方式，选用SVM算法检测，提示没有检测到Webshell，如图5-4

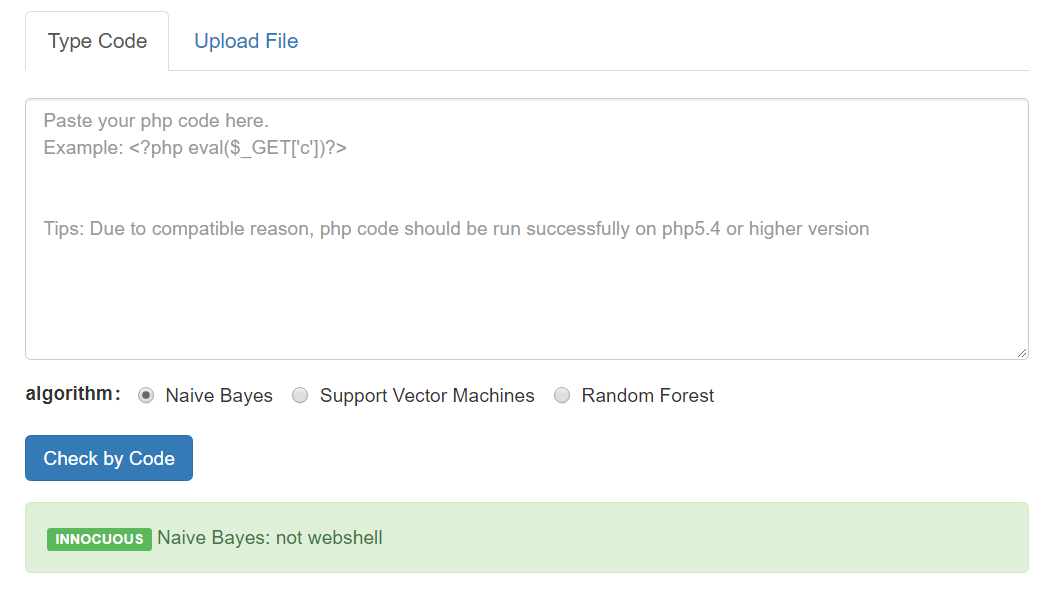


图5-4 正常单文件检测

（4）进行批量检测，提示检测结果，如图5-5

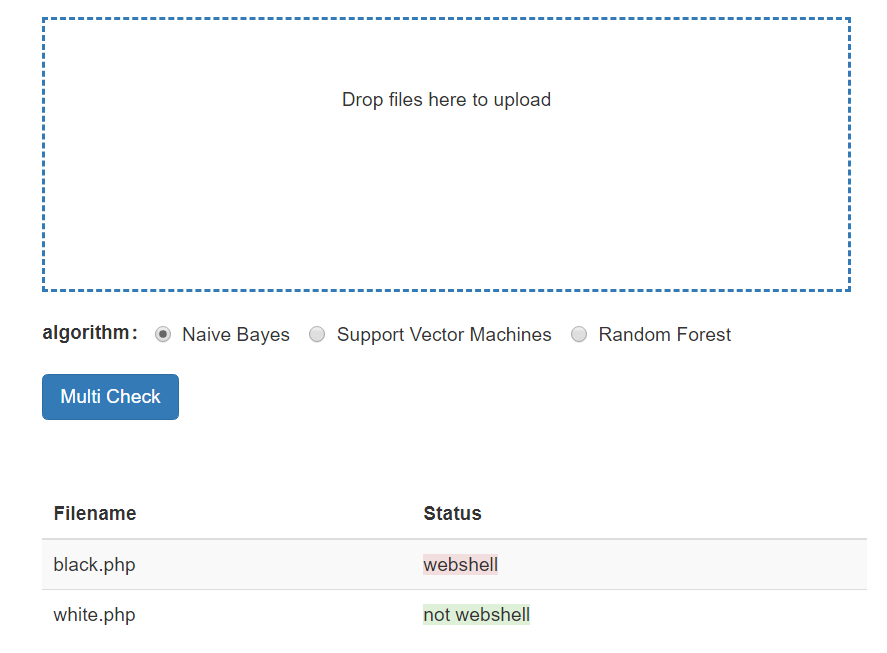


图5-5 批量检测

（5）调用API方式进行检测，如图5-6

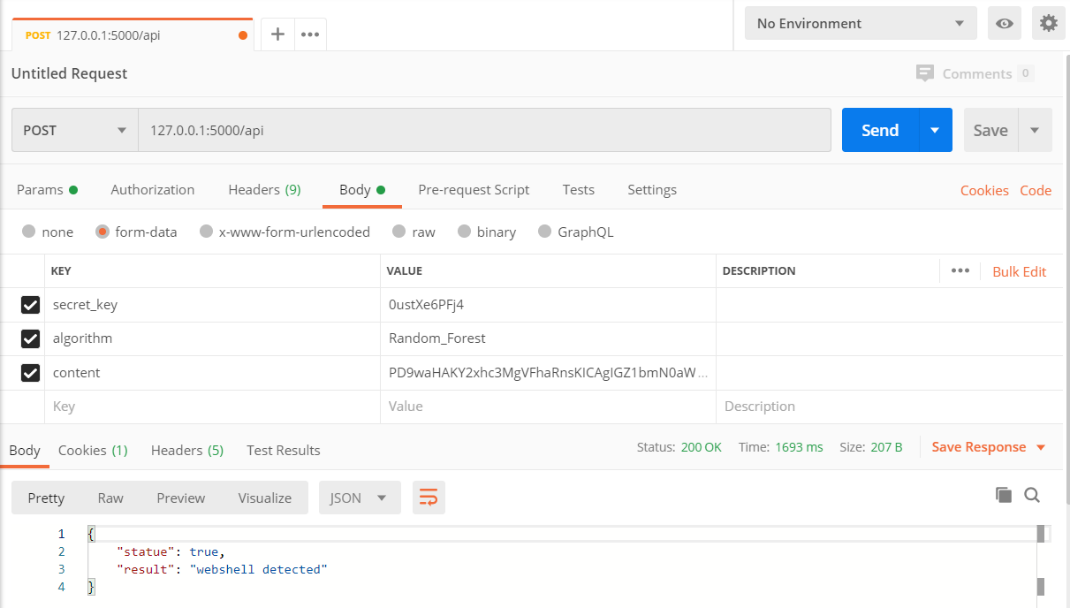


图5-6 API方式调用

（7）首页留言模块，点击提交后会在首页显示用户名，邮箱地址，留言内容等，如图5-7所示

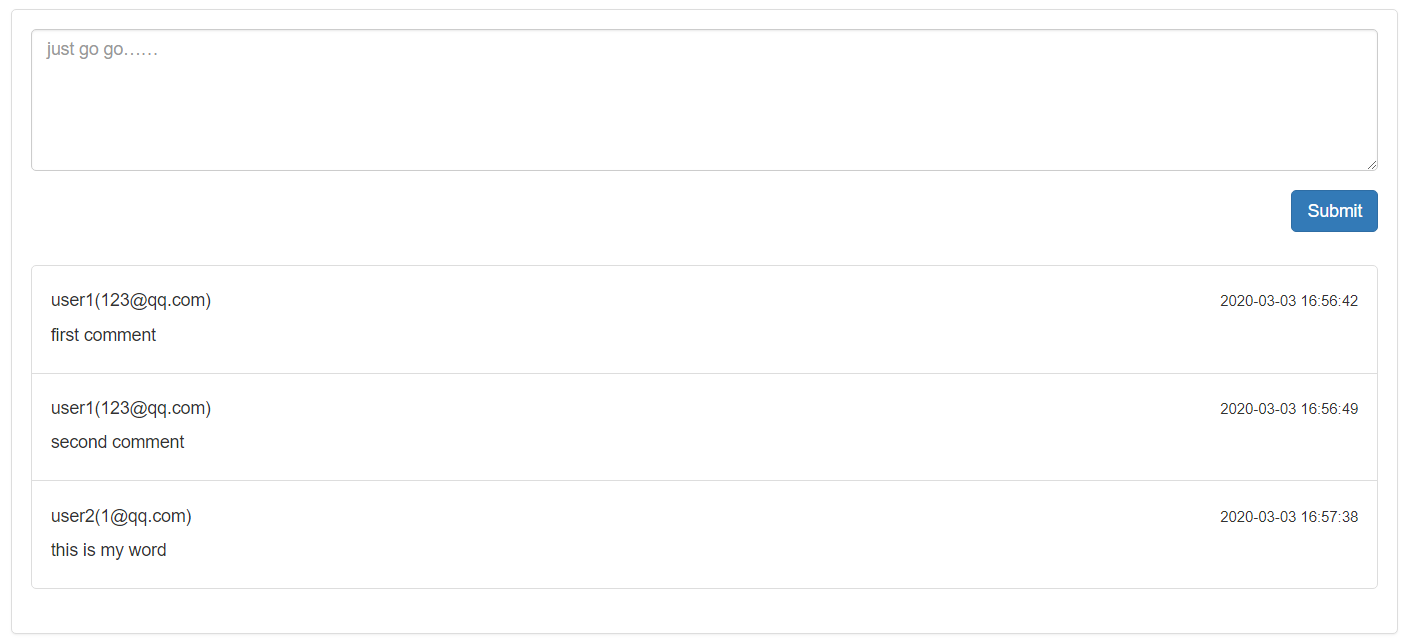


图5-7 留言模块

（8）API界面，如图5-8所示

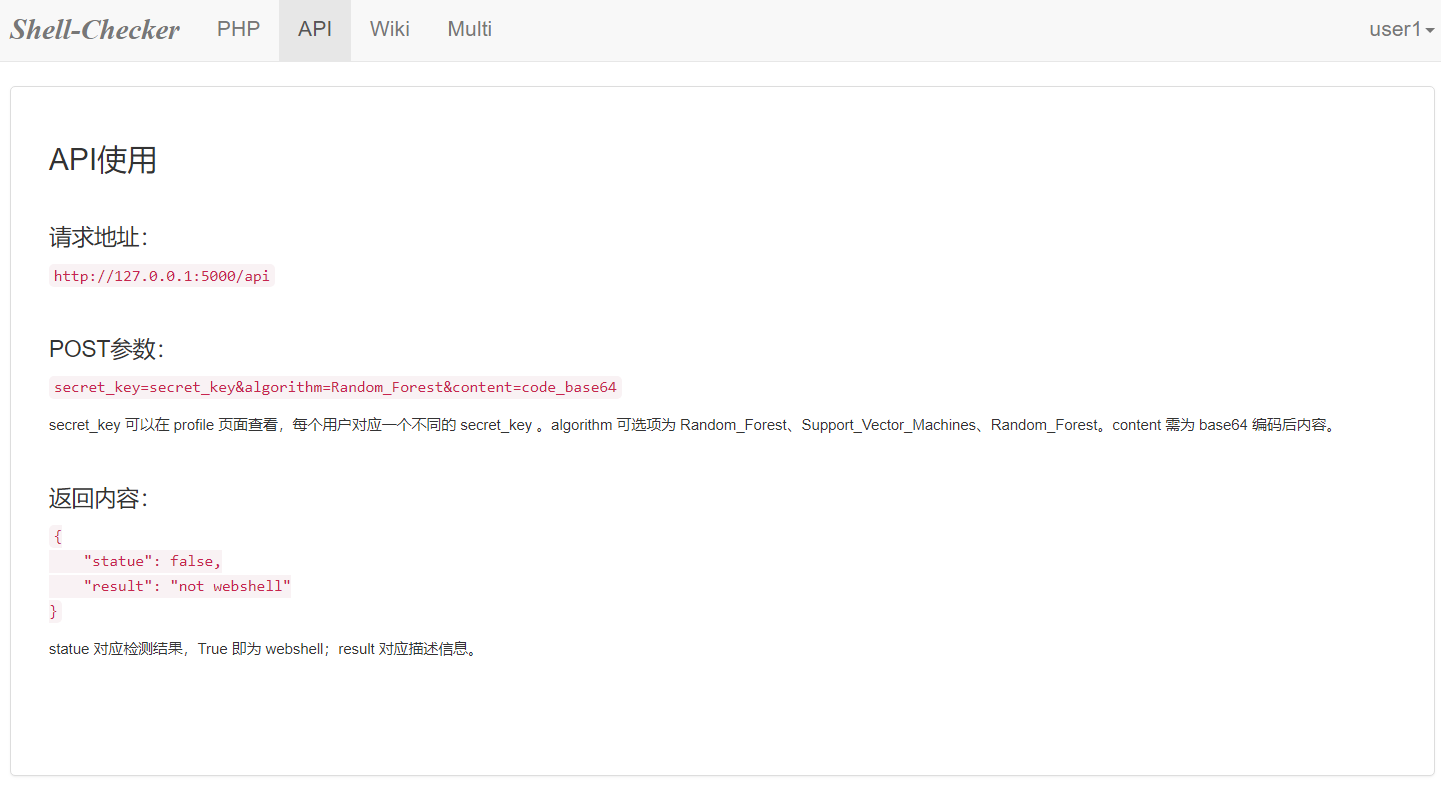


图5-8 API界面

（9）个人资料界面，可查看和修改个人资料，查看检测记录并修改检测结果，如图5-9

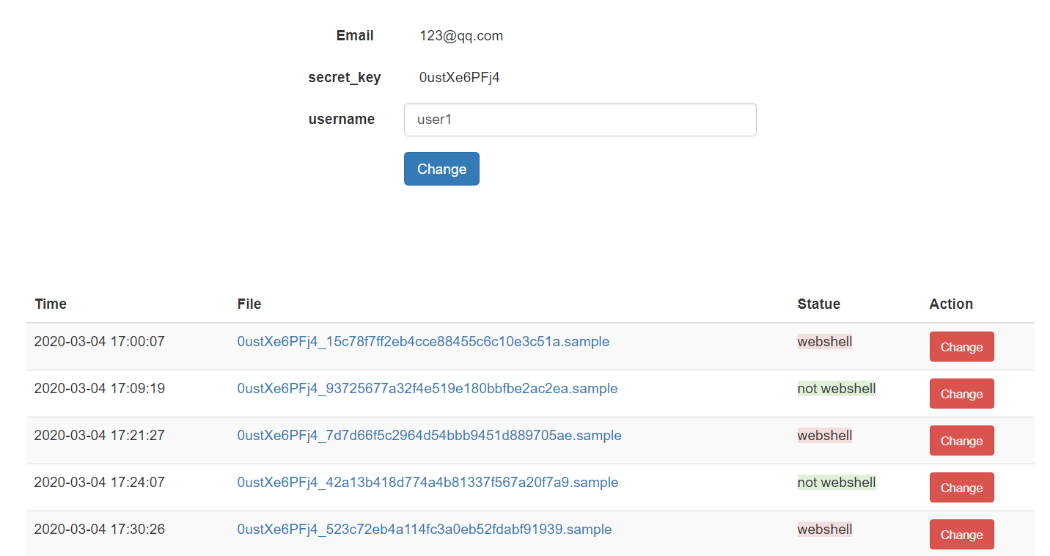


图5-9 个人资料界面

（10）后台Wiki页面管理，可修改Wiki页面显示内容，如图5-10

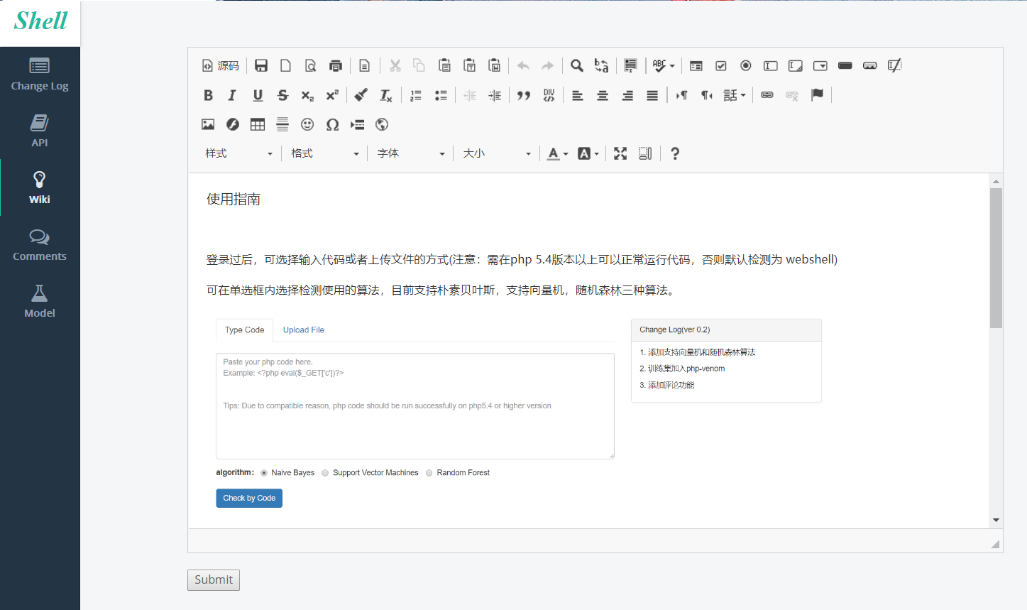


图5-10 后台Wiki页面管理

（11）后台评论管理，可查看评论以及删除评论，如图5-11

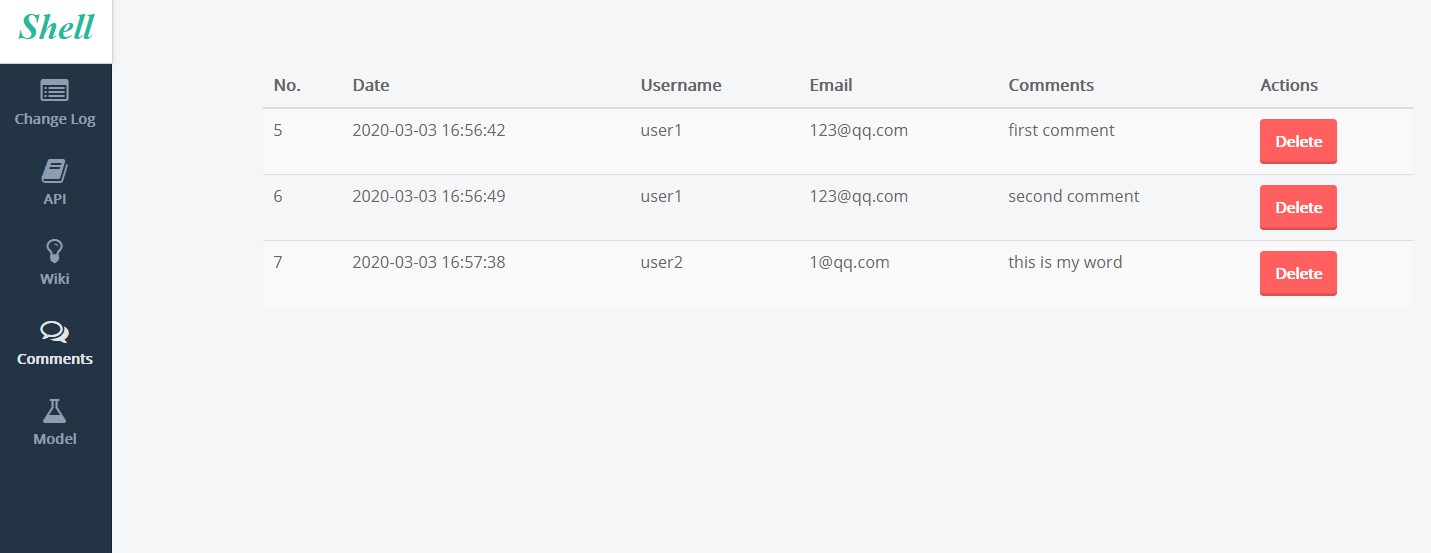


图5-11 后台评论管理

（12）后台模型管理，可重新对数据进行再训练，并选择算法使用的模型，如图5-12

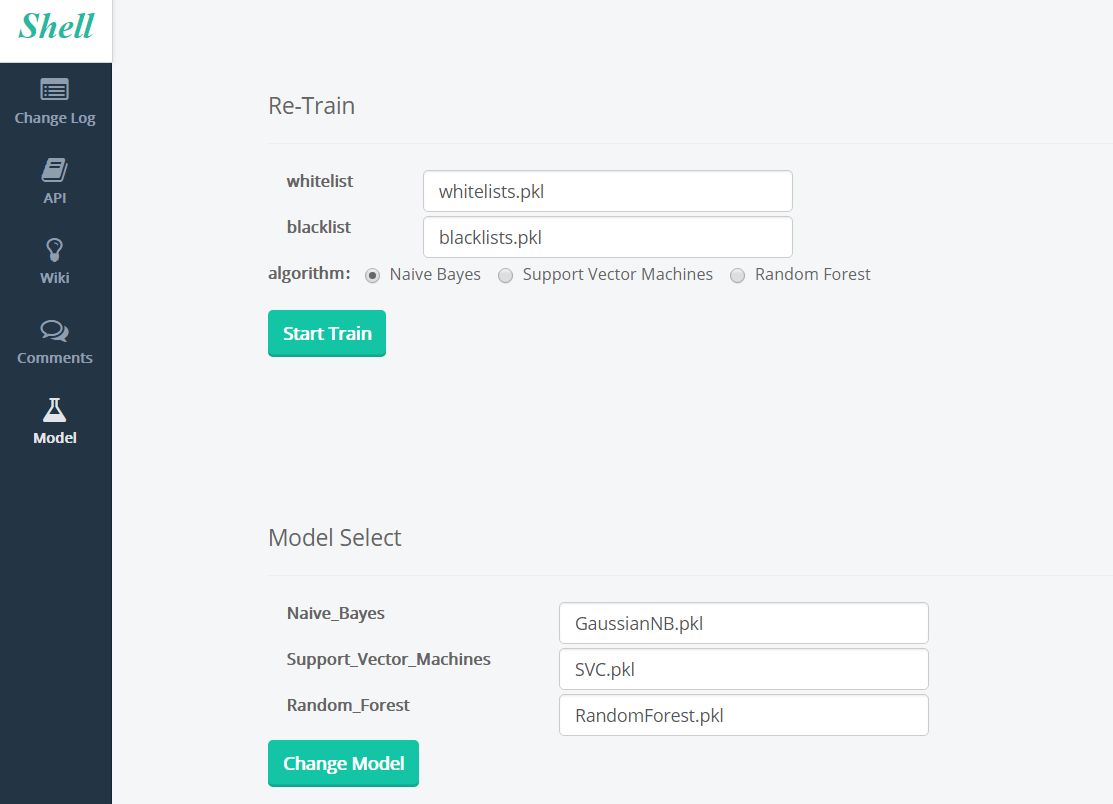


图5-12 后台模型管理

结 语

本次毕设我选择的是基于机器学习的恶意代码检测软件的开发，相对于传统机器学习检测系统只采用文本特征，本系统选择了PHP opcode与文本特征结合的方式，能够很好地避免注释以及新特性影响，提高了正确率。在设计和实现的工程了遇到了很多的知识盲点还有程序运行错误，但都在自己地耐心研究，查找资料以及老师和同学的帮助下克服了完成了一个完整的Flask Web项目。

在开发和研究的过程中，研究了许多关于Webshell的知识以及相关的检测手段，同时也学习到很多了关于机器学习的基础知识，同时也学习到了Flask开发的相关知识，对ORM的原理有了更深的理解，并且能够很好地应用，同时对Python Web开发的流程也更加熟练。长路漫漫，之后的学习工作生活还会有更多的挑战和新知识等着我去发现并学习，在这次毕业设计中我学会了仔细耐心，相信这些一定会对今后有所指导。

参考文献

1. 张贺威，刘晓洁.基于文本向量的php-webshell检测方法[J]. 数据通信,2019,16（4）:102-105.
2. 秦英.基于随机森林的WebShell检测方法[J]. 计算机系统应用.2019,28（18）:112-113.
3. 深信服科技股份有限公司.WEBSHELL后门识别方法、装置、设备及存储介质[P].中国. CN201910446470.3.2019.
4. 杭州安恒信息技术股份有限公司. 一种WebShell检测方法、装置、设备及介质[P].中国. CN201911094291.4.2019.
5. 关洪超. 基于HTTP流量的WebShell检测研究[D].北京:北京邮电大学[硕士论文],2019.
6. 潘杰. 基于机器学习的WebShell检测关键技术研究[D].天津:中国民航大学[硕士论文],2019.

致 谢

非常感谢王永丽老师对设计过程以及论文修改的悉心指导，她非常细心地解答我在过程中所遇到的问题以及纠正我在论文编写过程中的错误。在此向他表示我最衷心的感谢！

在过程当中本人还得到了张金全老师和崔海涛同学的热心支持和帮助。本人向他们表示深深的谢意！

同时也感谢抽出宝贵时间评审本文的各位专家、老师们！

作者简介：

姓 名：刘小川 性别：男

出生年月：1998年2月 民族：汉族

E-mail:str3am@qq.com

# 声 明

本论文的工作是2019年12月至2020年6月在成都信息工程大学网络空间安全学院完成的。文中除了特别加以标注地方外，不包含他人已经发表或撰写过的研究成果，也不包含为获得成都信息工程大学或其他教学机构的学位或证书而使用过的材料。

关于学位论文使用权和研究成果知识产权的说明：

本人完全了解成都信息工程大学有关保管使用学位论文的规定，其中包括：

（1）学校有权保管并向有关部门递交学位论文的原件与复印件。

（2）学校可以采用影印、缩印或其他复制方式保存学位论文。

（3）学校可以学术交流为目的复制、赠送和交换学位论文。

（4）学校可允许学位论文被查阅或借阅。

（5）学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

除非另有科研合同和其他法律文书的制约，本论文的科研成果属于成都信息工程大学。

特此声明！

作者签名：

日 期: 2020年6月 15日