



本科生课程论文



题 目 基于机器学习的 web 攻击检测

课 程 名 网络空间安全技术

任课教师 方勇

学 院 网络空间安全

专 业 网络空间安全

学生姓名 郭文博

学 号 2016141223014 年级 2016 级

目录

引言	4
课程概述	4
设计目的	4
系统开发环境	4
设计要求	5
系统分析与设计	5
功能需求说明	5
设计思路	5
总体设计	7
系统模块实现	8
数据加载模块	8
数据处理模块	8
数据降维模块	9
模型训练模块	10
攻击类型检测模块	11
URL 监测模块	12
URL 预测分析模块	13
可视化设计	14
UI 设计	14
功能实现	15
系统测试	15
功能模块测试	16
可视化模块测试	19
结束语	20
致谢	20
参考文献	21

基于机器学习的 web 攻击检测

郭文博

【摘要】随着搜索引擎的技术发展, web 技术也得到了长足的进步。目前中国的网站以及相关的博客等呈现出较高的增长率。而随之带来的就是各种各样的安全问题。比较常见的如 sql 注入, xss, csrf, 文件上传等。这些漏洞都是危害极大的。攻击者可以通过这些攻击受害者的网站以及相关的服务器, 获取其中的数据和重要的资源。而对于这类攻击的检测方法也是有很多的, 但是常规的就是基于特征库匹配的原理进行防御。该方法对于已知的攻击手段以及 payload 的效果比较好, 但是对于未知的攻击却无能无力。该项目就是尝试实现对 web 常见攻击的检测, 我们采用时下最为流行的机器学习的方式进行检测可能的攻击。对于机器学习的检测准确度比较的高。我们将该系统部署在主机上然后进行检测 80 端口的数据包, 对数据包中的 URL 进行提取, 然后使用训练好的模型进行判断其是正常的请求还是恶意的。达到自动检测攻击的目的。

【关键词】网络安全 ; 机器学习 ; web ; payload ;

1 引言

网络安全技术是网络空间安全基础必需课程。对于网络安全的相关的技术有一定的深入了解以及学习。而课程设计的题目就是实现基于机器学习的恶意 URL 检测。该项目的主要的功能是实现对请求的 URL 进行检测，判断其是正常的还是恶意的。如果是恶意的攻击就进行报警提示。对于机器学习的实现我们采用的是 SVM 和逻辑斯蒂回归两种方式进行训练模型。因为本身判断的结果就只有两种，要么为正常请求，要么是恶意的攻击。所以这里就采用这两种模型。然后我们使用大量的正常的 URL 和恶意 URL 进行训练模型。对于数据的来源我使用的是来自 Github 的数据。同时我们使用 scapy 库函数进行监听主机的 80 端口上的数据包。然后提取其中的 URL 请求，然后使用模型进行判断检测。如果是正常的请求就忽略，如果是恶意的请求就报警提示。以此来实现 web 攻击的动态的检测。

2 课程概述

2.1 设计目的

本项目是四川大学网络空间安全网络安全技术课程的实践内容。该项目主要的目的是实现基于机器学习的 web 攻击检测。在实现的过程中的要求就是自动的检测恶意的攻击，我们项目通过对请求的 URL 进行检测，然后分析其的负载进而判断其是不是正常。如果是正常的请求就忽略。如果是恶意的 URL 请求就进行报警提示。同时在项目中我们需要对攻击的类型进行判断，分析其属于 sql 注入攻击还是 xss 等攻击类型。在实现的过程中我采用的方式是特征匹配的方式进行判断。不同的攻击方式有不同的特征，所以根据这一点进行判断。同时在项目的实现过程中需要我们对机器学习的相关的模型以及各种攻击的方式熟悉。

2.2 系统开发环境

硬件平台：2.3 GHz Intel Core i5 8 GB 2133 MHz LPDDR3

操作系统：Mac os

支撑环境: python3, TensorFlow

其它有关组件: wireshark

2.3 设计要求

- 开发一套智能 Web 攻击检测系统;
- 实现对 Web 访问语义的解析;
- 实现基于机器学习的攻击检测引擎;
- 实现攻击信息获取与展示。

3 系统分析与设计

3.1 功能需求说明

1. Web 访问语义的解析功能: 功能的要求系统实现对于访问的 URL 的语义进行解析, 判断其是哪一种攻击方式。常见的 web 攻击包括 sql 注入, xss, csrf 等。
2. 基于机器学习的攻击检测引擎: 该功能的要求是实现对于恶意攻击的检测分析。本质上就是对于 URL 的判断, 分析其实正常的还是恶意的请求。
3. 实现攻击信息获取与展示: 该功能的要求就是实现一个可视化的界面, 实时的展示对于系统的检测结果。将接收到的恶意的 URL 访问展示出来, 提供报警提示的功能。

3.2 设计思路

本项目是基于机器学习的 web 攻击检测。所以在实现的过程中需要做的就是对于数据的获取以及处理, 同时还需要机器学习对数据进行训练, 使用得到的模型去检测访问的 URL 请求。因此项目中我们采用模块化设计的方案。将系统的总体设计分为数据的加载, 数据的处理, 数据降维, 模型的训练, 攻击类型判断以及 URL 提取和 URL 分析预测。下面一一介绍各个模块的功能以及设计

1. 数据加载模块

该模块的设计功能就是实现对训练数据的加载。在项目中我们的训练数据主要包括正常的 url 请求以及 xss payload, sql 注入的

payload, 和 csrf 攻击的 payload。我们需要将所有的训练数据读入到列表中供后续的训练模型使用。

2. 数据处理模块

该模块的设计主要用于处理训练的数据。在获取到的训练数据中，有很多的格式，比如说有的 URL 中包含大小写。有的 URL 采用的编码方式不一样。我们需要将这各类的因素全部排除。通过将所有的 url 转化为小写，同时采用一致的编码格式。这样就可以实现统一所有训练数据的格式，提高训练模型的准确度。

3. 数据降维模块

该模块的功能主要是实现数据的降维处理。在模型的训练中数据量比较的大，同时数据的特征比较的多，有很大一部分都是无用的特征。所以为了训练的效率以及准确度。这里我们采用 kmeans 进行降维处理。

4. 模型训练模块

该模块是项目的核心部分。主要的功能是使用机器学习训练一个用来检测 URL 请求的模型。对于机器学习的模型有很多的选择。比如支持向量机，逻辑斯蒂回归，卷积神经网络等等。但是在选择的过程中需要综合的考虑各方面的因素，比如准确度，效率，训练时间等等。

5. 攻击类型检测模块

该模块设计的目的就是用来检测攻击的类型。常见的攻击类型有 sql 注入，xss，csrf 等等。所以在模型中我们只是判断出其是正常请求还是恶意的请求，但是无法进一步判断其具体属于哪一种攻击类型。所以该模块就是在判断出恶意请求的基础上进一步判断其的类型。这里我们采用的方式就是基于特征匹配。不同的攻击方式必然有不同的特征，所以我们通过该方式就可以成功的识别其的具体攻击类型。

6. URL 监测模块

URL 监测模块主要是对于实际场景的考虑。在实际的运用过程中，

我们接收到的并不是一个个处理好的 URL。而是原始的数据包。在数据包中携带着请求的 URL。该模块的功能就是实现对数据包你的监控和处理，提取出我们需要分析的 URL。

7. URL 分析预测模块

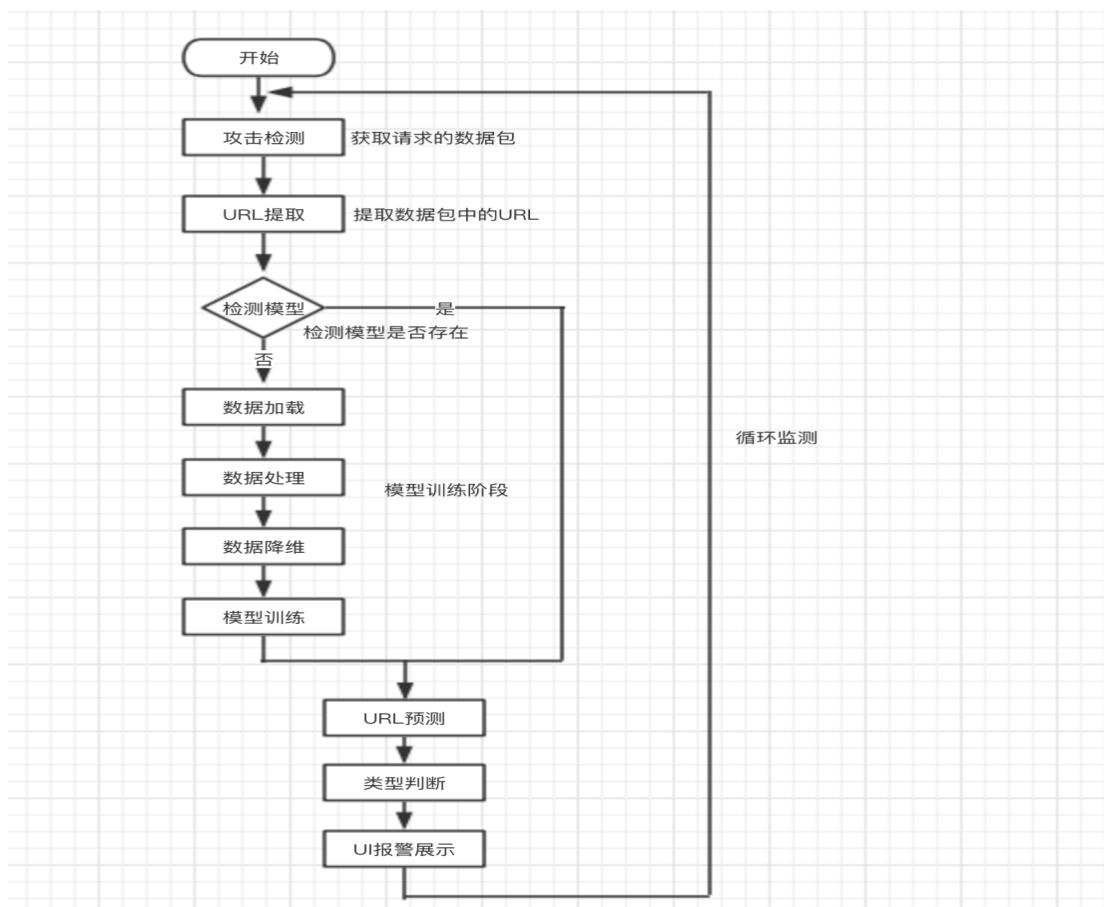
该模块的设计就是实现对上一步提取的 URL 的预测分析。调用机器学习的模型对 URL 进行检测，识别其是正常的请求还是恶意的攻击。同时识别其攻击的类型。

8. UI 模块

UI 模块的设计目的是实现监测的实时的报警和提示功能。系统会实时的检测 80 端口接收到的服务，同时模型分析结束之后，使用 UI 界面将攻击的具体信息展示出来。包括原目的 ip 地址，原目的 mac 地址以及攻击类型，代理类型和攻击的 payload。

3.3 系统总体设计

该项目的总体设计方案如下图所示：



项目总体设计图

4 系统模块实现

该部分是系统模块的具体实现方案以及代码。这里我们按照系统分析设计部分的设计然后来一一完成其功能。具体的模块设计实如下：

1. 数据加载模块

该模块的设计目的是实现训练数据的加载。在项目的实现中，我们使用的数据来自于 Github。数据的文件格式是以 TXT 形式保存的。所以在项目的第一步就是将这些数据读取出来然后保存在列表中以供后面的模块使用。该功能的实现比较的简单，具体代码如下所示：

```
# 从txt文件中获取测试数据
def get_url():

    good_url_list = []
    bad_url_list = []
    with open(good_url_train,'r') as good_url:

        for i in good_url.readlines()[:]:
            i.strip()
            good_url_list.append(i)

    with open(bad_url_train,'r') as bad_url:

        for i in bad_url.readlines()[:]:
            i.strip()
            bad_url_list.append(i)
    return [good_url_list, bad_url_list]
```

2. 数据处理模块

数据处理模块就是将第一步读取出来的数据进行处理。原始数据格式各有不同，同时编码方式也不同，大小写不统一。这些因素都会后续的文本特征的提取，进而影响模型的训练准确度。原始数据如下图所示：

/HR3/Personnel/Lzh/LzhIncomeEdit.aspx	oper=m&recid=101536 UNION ALL SELECT
NULL,NULL,NULL--	oper=m&recid=101536 UNION ALL SELECT
/HR3/Personnel/Lzh/LzhIncomeEdit.aspx	oper=m&recid=101536 UNION ALL SELECT
NULL,NULL,NULL,NULL--	oper=m&recid=101536 UNION ALL SELECT
/HR3/Personnel/Lzh/LzhIncomeEdit.aspx	oper=m&recid=101536 UNION ALL SELECT
NULL,NULL,NULL,NULL,NULL--	oper=m&recid=101536 UNION ALL SELECT
/HR3/Personnel/Lzh/LzhIncomeEdit.aspx	oper=m&recid=101536 UNION ALL SELECT
NULL,NULL,NULL,NULL,NULL,NULL--	oper=m&recid=101536 UNION ALL SELECT
/HR3/Personnel/Lzh/LzhIncomeEdit.aspx	oper=m&recid=101536 UNION ALL SELECT
NULL,NULL,NULL,NULL,NULL,NULL,NULL--	oper=m&recid=101536 UNION ALL SELECT

```
/1r8cfRLF.x?<meta http-equiv=set-cookie content="testydan=5106">
/main.php?logout=&uname\x09#
/en-us/dda2q7j.cfm?
/exchange/..%c1%8s..%c1%8s..%c1%8s..%c1%8s../winnt/system32/cmd.exe/?c+dir+c:
\+/og
/javascript/crontabs.exe
/help.php?q='|sleep\x0910\x09#
/l4fz1daw.php3?
/scripts/qbch5ojumj32.sh
/help.php?q='x0ddel q95327467 #
/scriptina2weblogsc.com/
```

通过数据处理模块我们的目的就是将格式统一，大小写统一，编码方式统一，同时我们也将数据中不重要的一些数据进行处理。消除其对于实验的影响。将 <http://www>. 将协议以及域名进行过滤。因为所有的数据中这都是一样的，对模型训练没有任何的帮助。具体的数据处理代码如下：

```
# 对数据进行处理
# 将所有的数据全部转化为小写
def deal_word():
    url = get_url()
    good_url_list = url[0]
    bad_url_list = url[1]
    for i in range(len(good_url_list)):
        deal_good_url = good_url_list[i].lower()
        deal_good_url = unquote(unquote(deal_good_url))
        good_url_list[i] = deal_good_url

    for i in range(len(bad_url_list)):
        deal_bad_url = bad_url_list[i].lower()
        deal_bad_url = unquote(unquote(deal_bad_url))
        bad_url_list[i] = deal_bad_url

    return [good_url_list, bad_url_list]

# 使用正则表达式进行匹配

def split_word(one_url):
    deal_word = []
    one_url=one_url.lower()
    one_url=unquote(unquote(one_url))
    one_url,num=re.subn(r'\d+\.\d+',one_url)
    one_url,num=re.subn(r'^(http|https)://[a-zA-Z0-9\.\:@#\!#\?\?]+\.' "http://u", one_url)
    r = '(?x)[\w.]+?(\|\|"\w+?"|\'\w+?'|http://\w|</\w+>|<\w+>|<\w+|\w+=|=|>|[\w.]+)'
    word_split = nltk.regexp_tokenize(one_url,r)
    for item in word_split:
        if item == '0' or item == 'http://u' or item == '':
            continue
        deal_word.append(item)
    return deal_word
```

3. 数据降维模块

数据降维模块的设计主要是降低训练集数据的维度。根据实验的效果发现，训练集的数据一共有 4000 多个维度。对于训练的难度比较的大。所以在这里我们进行数据的降维处理。在实验中我们发现对于数据

的维度大概 80 左右的效率比较的好。这里我们使用 kmeans 聚类算法进行降维。具体代码如下所示：

```
def kmeans(self, weight):
    """
    n_clusters: 指定K的值
    max_iter: 对于单次初始值计算的最大迭代次数
    n_init: 重新选择初始值的次数
    init: 制定初始值选择的算法
    n_jobs: 进程个数, 为-1的时候是指默认跑满CPU
    """
    clf = KMeans(n_clusters=num_clusters, precompute_distances=False, max_iter=300, n_init=40, init='k-means++')
    s = clf.fit(weight)
    print(s)

    # 保存聚类的结果
    self.label = clf.labels_
    print(self.label)
    with open('model/train.label', 'w') as output:
        for i in self.label:
            output.write(str(i) + '\n')

    return weight, self.label
```

4. 模型训练模块

攻击模型训练模块是实现对于 web 攻击检测模型的训练功能。在前几个模块中我们实现了对数据的加载以及处理清洗。所以下一步我们就是需要将这些数据进行训练。根据项目的性质我们发现实际上是一个二分类的情况。对于检测的 URL 要么是正常的请求要么是恶意的攻击。所以在实验中我们选择的训练的模型就是分类算法。分别采用逻辑斯蒂回归和支持向量机进行训练。在实验中我们分别对两种算法模型进行测试训练，选择准确度比较高的模型。同时在训练之前我们需要将数据进行向量化，因为模型接受的输入特定的数字矩阵。这里我们使用常见的 TD-IDF 进行向量化转化，TD-IDF 是一种用于资讯检索与文本挖掘的常用加权技术，被经常用于描述文本特征。TF 词频（Term Frequency），表示词条在某文档中出现的频率。IDF 逆向文件频率（Inverse Document Frequency），作用在于如果包含词条的文档越少，则 IDF 越大，说明该词条具有很好的类别区分能力。TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。

代码如下图：

```

def model_train(self):
    good_url_y = []
    bad_url_y = []

    for i in range(len(self.url_list[0])):
        good_url_y.append(0)

    for i in range(len(self.url_list[1])):
        bad_url_y.append(1)

    X = self.url_list[0]+self.url_list[1]
    Y = good_url_y + bad_url_y

    # 定义矢量化实例
    self.vectorizer = TfidfVectorizer(tokenizer=self.get_ngrams)
    """
    fit_transform()
    先拟合数据，再标准化
    """
    X = self.vectorizer.fit_transform(X)

    print(X)
    # 使用kmeans对原始的数据进行降维，以降低数据处理的复杂度
    weight, label = self.kmeans(X)
    X = self.transform(weight)

    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

    # self.lgs = LogisticRegression()
    self.lgs = svm.SVC()

    self.lgs.fit(X_train, y_train)
    self.lgs.score(X_test, y_test)

    print('模型的准确度:{}' .format(self.lgs.score(X_test, y_test)))

```

5. 攻击类型检测模块

攻击类型检测模型就是实现对攻击方式的判断。常见的 web 攻击方式有很多种，比如 sql 注入，xss 攻击以及文件上传和 csrf 等。同时在攻击的过程中攻击的载荷一般是夹杂在 URL 中，所以我们需要对 URL 进行分析，提取出其中的特征，然后匹配特征库中的特征字段，然后判断其对应的攻击等级以及危害程度。对于 URL 中的特征提取方法我采用的是传统的正则表达式匹配。

代码实现如下图所示：

```
# -*- coding: UTF-8 -*-
xss_high = ['<script>', '</script>', '<iframe>','</iframe>','response','write','eval','prompt','alert','javascript;']
xss_middle = ['Onclick','onerror','<!- ->','<base>','<base>>','location','hash','window','name','<form>','</form>']
xss_low = ['echo','print','href','sleep']

sql_high = ['and','or','xp_','substr','utl_','benchmark','shutdown','@version','information_schema','hex(']
sql_middle = ['select','if','union','group','by','count','/**/','char','drop','delete','concat','orderby','c']
sql_low = ['and','or','like','from','insert','update','create','else','exist','table','database','where','sleep']

def find_type(url_list_char):
    for i in url_list_char:
        if i in XSS_high:
            attack_rank = '攻击类型: XSS 攻击等级: 严重'
        elif i in XSS_middle:
            attack_rank = '攻击类型: XSS 攻击等级: 中级'
        elif i in XSS_low:
            attack_rank = '攻击类型: XSS 攻击等级: 轻微'
        elif i in SQL_high:
            attack_rank = '攻击类型: SQL注入 攻击等级: 严重'
        elif i in SQL_middle:
            attack_rank = '攻击类型: SQL注入 攻击等级: 中级'
        elif i in SQL_low:
            attack_rank = '攻击类型: SQL注入 攻击等级: 轻微'
        else:
            attack_rank = '攻击类型: 未知 攻击等级: 未知'

    return attack_rank
```

6. URL 监测模块

URL 监测模块就是实现实时的检测主机的 80 端口的数据包同时进行数据包的分析并将其中的 URL 信息进行提取出来。这里我们使用的是 scapy 库函数中的 sniff。对于该函数只是将原始数据包捕获。所以在下一步我们使用 scapy_http.http 函数进行解析数据包。将各个层的数据分离出来。一般的攻击方式都是通过使用 payload 进行攻击的，所以在这里我们就必须进行数据包的处理。Payload 是处于应用层的数据，所以我们需要将其中的 host 以及 path 提取出来。原始数据包如下图：

scapy.http 解析数据如下图所示：

```
/Users/apple/anaconda3/envs/TensorFlow/bin/python3.6 /Volumes/Study/2018大三上课程/网络安全技术/课设/WAF/geturl.py
###[ Ethernet ]###
dst      = 58:69:6c:4c:47:53
src      = 8c:85:90:cc:3e:59
type     = 0x800
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 64
id       = 0
flags    = DF
frag     = 0
ttl      = 64
proto    = tcp
chksum   = 0x7da5
src      = 10.132.2.158
dst      = 111.10.64.231
\options \
###[ TCP ]###
sport    = 55352
dport    = http
seq      = 4158989125
ack      = 0
dataofs  = 11
reserved = 0
flags    = S
window   = 65535
checksum = 0xed52
urgptr   = 0
options   = [ ('MSS', 1460), ('NOP', None), ('WScale', 6), ('NOP', None), ('NOP', None), ('Timestamp', (636908268, 0)), ('SAckOK', b''), ('EOL', None) ]
###[ Ethernet ]###
dst      = 8c:85:90:cc:3e:59
src      = 58:69:6c:4c:47:53
type     = 0x800
```

代码如下图所示：

```
# This import works from the project directory
import scapy_http.http as http
except ImportError:
    # If you installed this package via pip, you just need to execute this
    # from scapy.layers import http

def Sniffer():
    rule = 'tcp port 80'

    # prn = lambda x: x[IP].src
    # lambda x: x.summary()
    # <script>alert(1)</script>

    http_sniffer = sniff(filter=rule, iface='en0', prn=pack_callback, count=10)

def pack_callback(packet):
    packet.show()
    pack_for_url = []
    if 'TCP' in packet:
        Ether_dst = 'Mac_Dst: ' + packet.dst
        Ether_src = 'Mac_Src: ' + packet.src
        IP_src = 'IP_Src: ' + packet.payload.src
        IP_dst = 'IP_Dst: ' + packet.payload.dst

        if packet.haslayer(http.HTTPRequest):
            http_header = packet[http.HTTPRequest].fields
            http_method = 'Method: ' + http_header['Method'].decode()
            http_agent = 'User-Agent: ' + http_header['User-Agent'].decode()
            http_host = 'Host: ' + http_header['Host'].decode()
            http_path = 'Path: ' + http_header['Path'].decode()
            deal_good_url = 'URL: ' + unquote(http_header['Host'].decode() + http_header['Path'].decode())
```

7. URL 分析预测模块

URL 分析预测模块的设计目的就是要将从 URL 检测模块提取出来的 URL 进行分析预测。分析预测主要是分部分，第一步就是实现 URL 性质的预测，判断其是正常的请求还是恶意的攻击。第二步就是在第一步的基础上进行攻击类型的判断。在实现的方式上有两种：第一种就是训练对应的机器学习模型进行预测，第二种方式就是使用特征匹配的方式进行类型判断。在项目的实现中我采用的是特征匹配的方式进行判断的。主要的原因是因为其简单，不需要训练。同时可以实现攻击等级的判

断。不同的攻击手段在 payload 中有不同的特征体现，所以在这里我们就使用特征匹配的方式进行分析。

代码如下图：

```
def predict(self,new_url):
    # 加载训练好的模型
    try:
        with open(model,'rb') as f:
            svm_model = pickle.load(f)
    # 如果模型不存在，则进行训练
    except FileNotFoundError:
        lgs,label = self.model_train()

    # 对数据进行处理
    # new_url = new_url.lower()
    for i in range(len(new_url)):
        new_url[i] = new_url[i].lower()
        new_url[i] = unquote(new_url[i])

    # new_url =unquote(new_url)

    self.vectorizer = TfidfVectorizer(tokenizer=self.get_ngrams)

    url_predict = self.vectorizer.fit_transform(new_url)

    url_predict = self.transform(url_predict.tolil().transpose(),label)

    res = lgs.predict(url_predict)

    if res == 0:
        print("url为正常请求")
    else:
        print("恶意url")
    print(res)
```

5 可视化设计

5.1 UI 设计

UI 设计的目的就是实时的展示攻击的信息。这里我们设计的就是将系统捕获的数据包中的 url 解析出来，然后使用模型进行预测分析，将预测分析的结果展示在 UI 界面上，用户对于系统的整体的安全状态以及受到的攻击可以有一个更加直观的理解。这里因为本次的课程实践一共是两个，第一个项目实现过一个 UI 界面。所以这里的 UI 界面界面就是在此基础上进行修改的。

代码实现：

```

def __init__(self, parent_init_name):
    self.parent_init_name = parent_init_name
    self.ListIP = []
    self.ListInformation = []

def set_init_windows(self):
    self.parent_init_name.title("web入侵检测系统")
    self.parent_init_name.geometry('700x700+10+10')
    self.parent_init_name.resizable(width=True, height=True)

    self.ImageFrame = Frame(self.parent_init_name)
    self.ImgFrame = LabelFrame(self.ImageFrame, text="Warning!!!")
    self.img = Image.open("../image/warning.jpeg")
    self.img = ImageTk.PhotoImage(self.img)
    Label(self.ImgFrame, width=627, height=220, image=self.img).grid(row=0, column=0, columnspan=3)
    self.ImgFrame.pack()
    self.ImageFrame.pack()

    self.textframe = Frame(self.parent_init_name)
    self.developer = LabelFrame(self.textframe, text="开发信息")
    self.developer.pack(padx=10, pady=10)
    self.textframe.pack()

    Label(self.developer, width=70, height=2, bg="white", text="开发人员: 郭文博\n").grid(column=0, row=0)
    Label(self.developer, width=70, height=2, bg="white", text="开发环境: Mac os\n").grid(column=0, row=1)
    Label(self.developer, width=70, height=2, bg="white", text="开发时间: 2018-1-05\n").grid(column=0, row=2)
    Label(self.developer, width=70, height=2, bg="white", text="版本号: D-1.00\n").grid(column=0, row=3)

# 扫描的局域网主机ip信息
self.IpFrame = Frame(self.parent_init_name)
self.iplistframe = LabelFrame(self.IpFrame, text="实时入侵检测结果显示")

```

5.2 功能实现



6.系统测试

6.1 功能模块测试

1. 数据加载模块

测试数据: ' data//badqueries.txt'

' data//goodqueries.txt'

```
<base href=data:/,-alert(1)/>
<svg><set href=#script attributeName=href to=data:,alert(1)/>
"/>.<<img src=x onerror=alert(1)//>&lt;&gt;
"><<img src=x onerror=alert(1);//>;
"text </script><script>alert(1)</script>">
<marquee loop=1 width=0 onfinish=1/confirm`/1/'>0</marquee>
<marquee loop=1 width=0 onfinish=confirm(1)>0</marquee>
<abeon style=font-size:12px onmouseover=confirm(1)>abeon
<svg onload=location='//p0.al'>
7D%3B%20if%20%5B%209%20-ne%20%24%7Bstr1%7D%20%5D%3B%20then%20sleep%200%3B%20else%20sleep%201%3B%20fi%20%252F%252F\n',
Process finished with exit code 0
```

2. 数据处理模块

目的: 大小写统一, 编码方式统一, http://www...域名去除

```
<div style=width:1px;filter:glow onfilterchange=alert(1)>x</div>
</> style=x:expression\28write(1)\29>
<form><button formaction="javascript:alert(1)">x</button>
<event-source src="event.php" onload="alert(1)">
<a href="javascript:alert(1)"><event-source src="data:application/x-dom-event-stream,event:click
data:xxx
" /></a>
<script<{alert(1)}></script >
<?xml-stylesheet type="text/css"?><!doctype x system "test.dtd"><x>&x;</x>
<?xml-stylesheet type="text/css"?><root style="x:expression(write(1))"/>
<?xml-stylesheet type="text/xsl" href="#?"><img xmlns="x-schema:test.xdr"/>
```

```
return deal_word
url_list = ['http://www.baidu.com/system/login.php?site_path=<script>alert(1)</script>', 'http://www.nmap.org/index.php?op=viewarticle&articleid=9999/**/union/**/select/**/1331908730,1,1,1,1,1,1--&blogid=1
or i in url_list:
    print(i)
    split_word(i)
   统计词频
split_word()

train_url
/Users/apple/anaconda3/envs/TensorFlow/bin/python3.6 /Volumes/Study/2018大三上课程/网络安全技术/课设/WAF/train_url.py
↑ http://www.baidu.com/system/login.php?site_path=<script>alert(1)</script>
↓ ['http://u', '_path=', '<script>', 'alert()', '0', ')', '</script>']
http://www.nmap.org/index.php?op=viewarticle&articleid=9999/**/union/**/select/**/1331908730,1,1,1,1,1,1--&blogid=1
['http://u', 'viewarticle', 'articleid=', '0', 'union', 'select', '0', '0', '0', '0', '0', '0', '0', 'blogid=', '0']
http://www.chengdu.com/main.php?id='onclick=alert(1)#
['http://u', 'onclick=', 'alert()', '0', ')']

Process finished with exit code 0
```

3. 数据降维模块

降维之前的维度为: (53953, 4162)

```
(53952, 3140) 0.18780305992681806
(53952, 2238) 0.15172589451336768
(53952, 921) 0.24208673095475955
(53952, 2254) 0.10894492255657595
(53952, 2950) 0.2839781278747257
(53952, 1778) 0.21030978971572617
(53952, 3169) 0.2269651144597714
(53952, 2246) 0.10642138642433997
(53952, 2425) 0.1289986224722196
(53952, 3127) 0.20287227893038232
(53952, 885) 0.13539958828425083
(53952, 3053) 0.18729193841483696
(53952, 847) 0.16020409727595508
(53952, 2368) 0.16453317276359725
(53952, 3072) 0.2119584018396751
(53952, 3129) 0.14941771509441712
(53952, 3401) 0.22923580052518966
(53952, 1738) 0.18077388157765825
(53952, 1718) 0.12663639198287013
(53952, 2593) 0.16490762767822598
(53952, 2905) 0.23156770009200434
(53952, 70) 0.1971908207589923
(53952, 375) 0.26897922993111684
(53952, 3582) 0.23710573528427692
(53952, 390) 0.2498216155844093
向量化后维度: (53953, 4162)
```

降维完成之后的维度为: (53953, 80)

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=80, n_init=40, n_jobs=None, precompute_distances=False,
       random_state=None, tol=0.0001, verbose=0)
[0 0 0 ... 0 0]
kmeans 完成,聚成 80类
降维后的维度: (53953, 80)
(38853, 0) 0.05975763511032674
(39395, 0) 0.05975763511032674
(53145, 0) 0.1219676772350814
(53805, 0) 0.1219676772350814
(52264, 0) 0.33111526879850445
(52264, 0) 0.33111526879850445
(38852, 0) 0.061156295393708135
(38853, 0) 0.05975763511032674
(39395, 0) 0.05975763511032674
(39396, 0) 0.061156295393708135
(51573, 0) 0.26488671315957846
(51846, 0) 0.15184593113876843
(52366, 0) 0.12200555456824116
(52536, 0) 0.20656628301427335
(52666, 0) 0.12900265296840274
(52698, 0) 0.10220412923654713
(52740, 0) 0.0924599850966213
(53752, 0) 0.12200555456824116
(53753, 0) 0.10220412923654713
(53802, 0) 0.20656628301427335
(53803, 0) 0.12900265296840274
(53806, 0) 0.0924599850966213
```

4. 模型训练模块

训练模型完成，准确度为 99%

```
(53818, 43) 0.2354922335523712
(53819, 43) 0.13362637151455015
(53852, 43) 0.3973483557584175
(53853, 43) 0.41996561129723337
(53854, 43) 0.3655045524022797
(53855, 43) 0.43059797785814463
(53856, 43) 0.22558876114018636
(53857, 43) 0.4453928159437263
(53858, 43) 0.4808482123082861
(53859, 43) 0.4808482123082861
(53860, 43) 0.4808482123082861
(53861, 43) 0.5747507266842118
(53862, 43) 0.48717575204759384
(53863, 43) 0.4291494313081377
(53864, 43) 0.42878370012415273
(53865, 43) 0.44606983666214184
(53866, 43) 0.4455106000698843
(53867, 43) 0.4527124294622283
(53868, 43) 0.36210782071963515
(53869, 43) 0.3772839862431062
(53870, 43) 0.43984600984542194
(53871, 43) 0.3859379387283411
(53872, 43) 0.3817153873268514
(53873, 43) 0.3741369471620094
模型的准确度:0.9960151978500602
```

5. 攻击类型检测模块

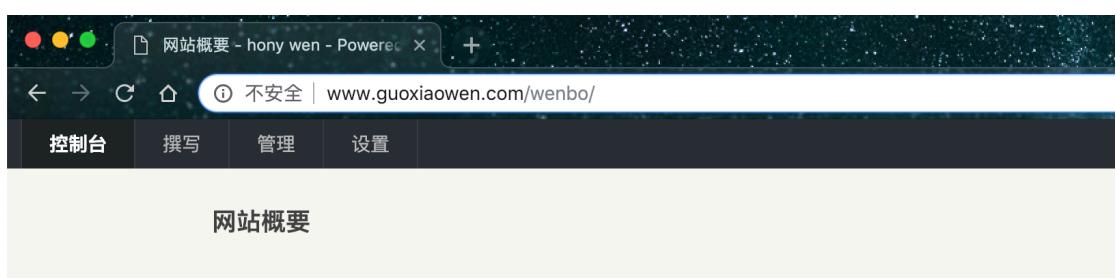
检测的部分结果如下:

```
['currentstudies']
攻击类型: 未知 攻击等级: 未知
/emfilefile2379/

['emfilefile0']
攻击类型: 未知 攻击等级: 未知
/hr3/personnel/lzh/LzhReportEdit.aspx recid=101533') AND (SELECT 8692 FROM(SELECT COUNT(*),CONCAT(0x3a7963643a,(SELECT (CASE WHEN (8692=8692) THEN 1 ELSE 0 END)) 
['hr0', 'personnel', 'lzh', 'lzhreportedit.aspx', 'recid=', ')', 'and', 'select', 'from', 'select', 'count(', ')', 'concat(', '0x0a0a', 'select', 'case', 'when', 
攻击类型: SQL注入 攻击等级: 严重
/ hr3/personnel/lzh/LzhReportEdit.aspx recid=101533' AND (SELECT 8692 FROM(SELECT COUNT(*),CONCAT(0x3a7963643a,(SELECT (CASE WHEN (8692=8692) THEN 1 ELSE 0 END)), 
['hr0', 'personnel', 'lzh', 'lzhreportedit.aspx', 'recid=', 'and', 'select', 'from', 'select', 'count(', ')', 'concat(', '0x0a0a', 'select', 'case', 'when', '0=',
攻击类型: SQL注入 攻击等级: 严重
/ hr3/personnel/lzh/LzhReportEdit.aspx recid=101533% AND (SELECT 8692 FROM(SELECT COUNT(*),CONCAT(0x3a7963643a,(SELECT (CASE WHEN (8692=8692) THEN 1 ELSE 0 END)), 
['hr0', 'personnel', 'lzh', 'lzhreportedit.aspx', 'recid=', 'and', 'select', 'from', 'select', 'count(', ')', 'concat(', '0x0a0a', 'select', 'case', 'when', '0=',
攻击类型: SQL注入 攻击等级: 严重
/ hr3/personnel/lzh/LzhReportEdit.aspx recid=101533' AND 8173=CAST((CHR(58)||CHR(121)||CHR(99)||CHR(100)||CHR(58))||(SELECT (CASE WHEN (8173=8173) THEN 1 ELSE 0
```

6. URL 监测模块

测试 URL:



捕获的结果:

```
49
pack_callback()

Run: geturl(1)
/Users/apple/anaconda3/envs/TensorFlow/bin/python3.6 /Volumes/Study/2018大三上课程/网络安全技术/课设/WAF/code/geturl.py
IP_Src: 10.132.2.158
IP_Dst: 149.28.21.190
Mac_Src: 8c:85:90:cc:3e:59
Mac_Dst: 58:69:6c:4c:47:53
URL: www.guoxiaowen.com/wenbo/
Method: GET
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36
Host: www.guoxiaowen.com
Path: /wenbo/
['URL': 'www.guoxiaowen.com/wenbo/', 'Method': 'GET', 'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36']

Process finished with exit code 0
```

7. URL 分析预测模块

```
Run: train_url(1) ×
  /skypearl/cn/validatorAction.action?d=1497630338673
  ↓
  loading model success
  url为正常请求
  /skypearl/cn/validatorAction.action?d=1497633730
  ↓
  loading model success
  url为正常请求
  /hr3/personnel/lzh/LzhReportEdit.aspx recid=101533' AND 8173=CAST((CHR(58)||CHR(121)||CHR(99)||CHR(100)||CHR(58))||(SELECT (CASE WHEN (8173=8173) THEN 1 ELSE 0 END))::)
  ↓
  loading model success
  恶意url
  /hr3/personnel/lzh/LzhReportEdit.aspx recid=101533' AND 8173=CAST((CHR(58)||CHR(121)||CHR(99)||CHR(100)||CHR(58))||(SELECT (CASE WHEN (8173=8173) THEN 1 ELSE 0 END))::)
  ↓
  loading model success
  恶意url
  /hr3/personnel/lzh/LzhReportEdit.aspx recid=101533' AND 9038=CONVERT(INT,(CHAR(58)+CHAR(121)+CHAR(99)+CHAR(100)+CHAR(58)+(SELECT (CASE WHEN (9038=9038) THEN CHAR(49)
  ↓
  loading model success
```

6.2 可视化模块测试

测试 URL : [http://www.solarpeng.com/?cat=<script>alert\(1\)</script>](http://www.solarpeng.com/?cat=<script>alert(1)</script>)

实时入侵检测结果显示

```
IP_Dst: 149.28.233.229
Mac_Src: 8c:85:90:cc:3e:59
Mac_Dst: 28:2c:b2:08:25:01
URL: http://www.solarpeng.com/?cat=<script>alert(1)</script>
Method: GET
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36 (KHTML, like
Host: www.solarpeng.com/
Path: /?cat=<script>alert(1)</script>
url为恶意攻击
攻击类型: XSS 攻击等级: 严重
```

[开始](#) [退出](#) [帮助](#)

测试结果正确

测试 URL : http://www.solarpeng.com/?cat=and exist 20select * from solarpeng

实时入侵检测结果显示

```
IP_Dst: 149.28.233.229
Mac_Src: 8c:85:90:cc:3e:59
Mac_Dst: 58:69:6c:4c:47:53
URL: www.solarpeng.com/?cat=and exists select * from solarpeng
Method: GET
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36 (KHTML, like
Host: www.solarpeng.com
Path: /?cat=and%20exists%20select%20*%20from%20solarpeng
url为恶意攻击
攻击类型: SQL注入 攻击等级: 轻微
```

[开始](#) [退出](#) [帮助](#)

测试结果正确

测试 URL : <http://www.solarpeng.com/>

```

- 实时入侵检测结果显示 -
IP_Src: 10.132.30.109
IP_Dst: 149.28.233.229
Mac_Src: 8c:85:90:cc:3e:59
Mac_Dst: 58:69:6c:4c:47:53
URL: www.solarpeng.com/
Method: GET
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36 (KHTML, like
Host: www.solarpeng.com
Path: /
url为正常请求

```

开始 退出 帮助

测试结果正确

6.2 模型对比测试

SVM 模型：

```

Run: train_url (1) ×
(53855, 43) 0.43059797785814463
(53856, 43) 0.22558876114018636
(53857, 43) 0.4453928159437263
(53858, 43) 0.4808482123082861
(53859, 43) 0.4808482123082861
(53860, 43) 0.4808482123082861
(53861, 43) 0.5747507266842118
(53862, 43) 0.48717575204759384
(53863, 43) 0.4291494313081377
(53864, 43) 0.42878370012415273
(53865, 43) 0.44606983666214184
(53866, 43) 0.445510600698843
(53867, 43) 0.4527124294622283
(53868, 43) 0.36210782071963515
(53869, 43) 0.3772839862431062
(53870, 43) 0.43984600984542194
(53871, 43) 0.3859379387283411
(53872, 43) 0.3817153873268514
(53873, 43) 0.3741369471620094
模型的准确度: 0.9960151978500602
恶意url

```

逻辑斯蒂回归模型：

```

Run: train_url (1) ×
(53854, 43) 0.3655045524022797
(53855, 43) 0.43059797785814463
(53856, 43) 0.22558876114018636
(53857, 43) 0.4453928159437263
(53858, 43) 0.4808482123082861
(53859, 43) 0.4808482123082861
(53860, 43) 0.4808482123082861
(53861, 43) 0.5747507266842118
(53862, 43) 0.48717575204759384
(53863, 43) 0.4291494313081377
(53864, 43) 0.42878370012415273
(53865, 43) 0.44606983666214184
(53866, 43) 0.445510600698843
(53867, 43) 0.4527124294622283
(53868, 43) 0.36210782071963515
(53869, 43) 0.3772839862431062
(53870, 43) 0.43984600984542194
(53871, 43) 0.3859379387283411
(53872, 43) 0.3817153873268514
(53873, 43) 0.3741369471620094
模型的准确度: 0.998795292373274

```

7 结束语

通过几周的努力,终于成功的完成了网络安全技术的课程设计. 本次的课程设计是基于机器学习的 web 攻击检测, 对于这个课题, 在刚开始的时候还是一脸茫然, 完全不知道如何下手。虽然自己之前也有学习过关于机器学的相关知

识，但是也不是很熟悉。同时对于 web 攻击的方式多种多样不知道该如何去检测。但是在后来的实现中慢慢的理到了其中的道理。同时该课题要求我们熟练的掌握关于机器学习相关的模型，同时在代码实现的过程中我们也要运用 python 来进行代码的编写，也要求我们熟练的掌握关于 python 的相关语法，掌握相关的库函数 tkinter, TensorFlow, python-scapy 库函数等，同时在完成设计的过程中我们也遇到了好多的问题，比如在 UI 设计的时候关于图片的加载问题，无论如何写都是无法正常的加载图片，在进行 URL 预测的时候遇到了 shape 不统一的问题，都耗费了很多的时间。但是最后都一一解决了。在数据的解析过程中出现了解析错误的问题，但是在后来的完善中都得到了很好的解决。完成课程设计的过程中，主要按照模块化的设计方式进行完成，同时使用类来将各功能封装，最后将所有的代码整合起来。虽然在项目的实现过程遇到了很多的问题，但是最后还是成功的实现了项目的要求，自己的相关的理论知识以及技术水平有了一定的提升。

8 致谢

本次课程设计中遇到了很多的问题。最后都一一的解决了。在此感谢各位老师以及同学的帮助以及指导。同时在本次的设计过程中还要感谢本次课程设计所参考的各个文献以及博客的作者，以及一起参加课程设计的同班同学，虽然时间比较紧，任务比较重，但是大家都很努力的完成了各自的项目。衷心的感谢他们。相信在未来的学习生活中，我们也将相互帮助，共同进步。

参考文献

- [1] 杨晓峰, & 杨静宇. (2011). 基于机器学习的 Web 安全检测方法研究.
- [2] 施光莹, & 唐振民. (2014). 基于 Active SVM 算法的恶意网页检测技术研究.
- [3] 孙伟, 张凯寓, 薛临风, 徐田华, & Sun Wei^{3, 4}, Zhang Kaiyu^{1, 4}, Xue Linfeng^{1, 4}, Xu Tianhua^{2, , Beijing 100044 ;3. School of Electronics Information Technology Sun Yet-sen University, Guangzhou 510006;4. Key Laboratory of Information Technology {Sun Yet-sen University} , Ministry of Education, Guangzhou 510006}.}

- (2016). XSS 漏洞研究综述 - A Review on Cross-Site Scripting. 信息
安全研究, 2(12), 1068–1079.
- [4] 顾明昌, 王丹, 赵文兵, & 付利华. (2016). 一种基于攻击向量自动
生成的 X SS 漏洞渗透测试方法. 软件导刊, 15(7), 173–177.
- [5] 程书宝. (n. d.). 基于支持向量机的 Web 攻击检测技术.