基于机器学习的攻击检测系统

之前读到一片freebuf的文章中利用机器学习来进行XSS的检测[1]，从中得到启发，可以利用机器学习完成其他类型攻击的检测。

问题

- 攻击的特征如何进行的提取?
- 攻击数据集的获取
- 采用什么样的算法进行训练?

前期准备

文中是利用TF-IDF来进行的特征提取。

`字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。`

首先采用文中的方法进行特征提取，其次，在观察特征时，发现对于正常的请求其很大概率是可读的字符串，所以对请求进行分割，获取其单词构成。

数据集包括了三个部分，一部分是GitHub上的payload集合，另一部分是secrepo上的http.log数据作为正常请求。第三部分是HTTP DATASET CSIC 2010数据集，其中包括36000条正常请求和25000条恶意请求。

在选择算法上，首先利用文中使用的逻辑回归进行实验，然后选择其他常见的分类算法进行实验，如knn，svm，朴素贝叶斯等。

在这里，考虑到CNN网络对提取局部的特征有比较好的效果，将数据放进CNN网络中进行实验。

复现

首先对文章进行复现，复现中最麻烦的一步是数据的特征提取，这里无法做到和文中完全一致，利用我对TF-IDF的理解进行复现。

这里选用的数据集为：恶意数据为GitHub上的payload，正常数据为http.log中的请求。

```
def split_url(data_set,num = 3):
    data_str= []
    for s in data_set:
        s = s.strip()
        s = " ".join([s[i:i+num] for i in range(len(s)-2)])
        data_str.append(s)
    return data_str
```

在获得特征矩阵后，就可以选择适当的算法进行训练。

```
# 逻辑回归
# 98.9%
# lr = LogisticRegression()
# lr.fit(x_train,y_train)
# predictions = lr.predict(x_test)
# pre = lr
```

```
#朴素贝叶斯
# 准确率为：98.3%
nb = MultinomialNB()
nb.fit(x_train,y_train)
predictions = nb.predict(x_test)
```

```
# knn
# 准确率为：94.8%   n = 6 数据集 50000 速度很慢
ner = KNeighborsClassifier(n_neighbors=6).fit(x_train,y_train)
predictions = ner.predict(x_test)
pre = ner
```

再现

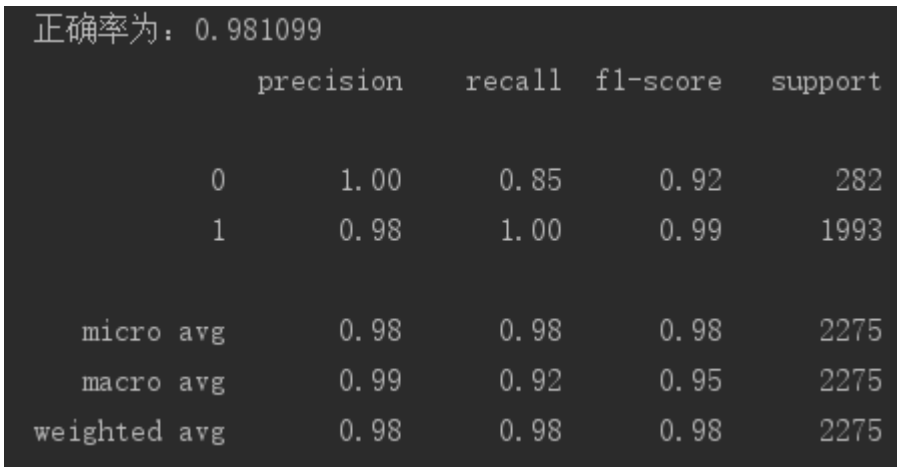
1、SQL注入检测

在复现完成后，实现对SQL注入的检测。

数据集选择：恶意数据为GitHub上的payload，大概有2700多条。正常数据为取20000条http.log中的数据。

步骤与上面类似，先进行数据特征的提取，计算特征矩阵，选择算法进行训练。

代码也与上面类似，这里不再赘述。



上图为使用逻辑回归的得分情况。

```
# 朴素贝叶斯
# 97.3%
# 逻辑回归
# 98 %
# knn
# 97.9%   n = 6
```

这里没有使用svm算法，因为其特征矩阵的维度十分高，svm的计算复杂度太高。

2、实现多类型的攻击检测

数据集选择使用HTTP DATASET CSIC 2010

1、TD-IF特征，机器学习算法实现

特征提取的方法不再使用TF-IDF中的n-gram，而是通过分词获得。

```
def parse_data(file_path):
    data_set = []
    #读取数据并且进行分割，保存
    with open(file_path) as f:
        lines_list = f.readlines()
        for s in lines_list:
            if s.startswith("GET") or s.startswith("POST"):
                s = s.split()[1][30:]
                s = re.split(r"[/.?%&=+]",s)
                s = " ".join(s)
                data_set.append(s)
    print(len(data_set))
    np.save("normal_traffic.npy",data_set)
```

分词前：

```
publico/anadir.jsp?
id=2&nombre=Jam%F3n+Ib%E9rico&precio=85&cantidad=%27%3B+DROP+TABLE+usuarios%3B+SELECT+*
+FROM+datos+WHERE+nombre+LIKE+%27%25&B1=A%F1adir+al+carrito
```

分词后：

```
publico anadir jsp id 2 nombre Jam F3n Ib E9rico precio 85 cantidad  27 3B DROP TABLE
usuarios 3B SELECT * FROM datos WHERE nombre LIKE  27 25 B1 A F1adir al carrito
```

```
#逻辑回归
```

尝试进行聚类分析，使用kmeans算法进行聚类，聚类失败。发现特征的维度太高，计算复杂度太高。

2、自编码特征，机器学习算法

上面在进行聚类时遇到的问题是数据的维度太高，是因为对请求进行分割，所造成的不同的字符串太多，在进行TF-IDF时得到的维度太高，这里尝试进行提取特定的特征。

结合所看文章和对请求的理解，提取了一下特征：

- URL长度
- 参数部分长度
- 参数的个数
- 参数的最大长度
- 参数的数字个数
- 参数值中数字所占比例
- 参数值中字母所占比例
- 特殊字符个数
- 特殊字符所占比例
- ~~相邻路径间的编辑距离~~

对于编辑距离，其计算的是相邻串的差异程度。这里再计算时会将顺序打乱，所以不将其作为特征。

```
for i in range(len(data_link)):
    per_fea = []
    url_len = len(data_link[i])
    per_fea.append(url_len)
    s = data_link[i].split("?")
    if len(s) != 1:
        par_len = len(s[1])
        par = s[1].split("&")
        par_num = len(par)
        par_max_l = 0
        number_num = 0
        str_num = 0
        spe_num = 0
        for pa in par:
            [par_name,par_val] = pa.split("=")
            if par_max_l < len(par_val):
                par_max_l = len(par_val)
            # pdb.set_trace()
            number_num = number_num + len(num_regex.findall(par_val))
            str_num = str_num + len(zimu_regex.findall(par_val))
            spe_num = len(par_val) - len(num_regex.findall(par_val)) -
len(zimu_regex.findall(par_val))
            number_rt =  number_num / len(par_val)
            str_rt = str_num / len(par_val)
            spe_rt = spe_num / len(par_val)
```

自编码的特征向量

```
array([18.        , 33.        ,  1.        , 24.        ,  0.        ,
        0.        , 23.        ,  0.95833333,  1.        ,  0.04166667])
array([22.        , 69.        ,  5.        , 16.        ,  3.        ,
        0.5       , 32.        ,  5.33333333,  0.        ,  0.        ])
```

选择算法进行训练

```
# 逻辑回归
# 正确率：73%
# 朴素贝叶斯
# 准确率：50.2%
# svm
# 正确率为：92.1%  ，速度较慢
# knn
# 正确率：90.9%   n = 6
```

可以看出提取的特征再逻辑回归和朴素贝叶斯上的效果都不好，但是svm和knn上效果不错，但svm的速度较慢，在knn上，速度很快，效果也不错。

3、卷积神经网络

考虑到卷积神经网络对局部特征的敏感性，这里利用CNN提取特征。

网络的构建

```
#初始化权重
def init_weight(shape,std_dev):
    weight = tf.Variable(tf.truncated_normal(shape,stddev= std_dev))
    return weight
#初始化偏置
def init_bias(shape,std_dev):
    bias = tf.Variable(tf.truncated_normal(shape,stddev=std_dev))
    return bias
#定义卷积层
def conv_2d(x,w):

    return tf.nn.conv2d(x,w,strides=[1,1,1,1],padding="SAME")
#池化层
def max_pool_2x2(x):
    return tf.nn.max_pool(x,ksize=[1,2,2,1],strides=[1,2,2,1],padding="SAME")
#全连接层
def fully_connected(input_layer,weights,biases):
    layer = tf.add(tf.matmul(input_layer,weights),biases)
    return tf.nn.tanh(layer)
```

```
#卷积模型
#第一层
shape_w = [10,3,1,32]
shape_b = [32]
weight_1 = init_weight(shape_w,std_dev=0.01)
bias_1 = init_bias(shape_b,std_dev=0.01)
layer1 = max_pool_2x2(tf.nn.relu(tf.add(conv_2d(x_data,weight_1),bias_1)))
```

这里再进行训练时发现数据的维度太高，无法进行训练，所以在训练之前对数据进行了降维处理。利用PCA算法对数据进行降维。这里保留数据98%的特征。

```
# # 对数据进行降维
# pca = PCA(n_components=0.98)
# pca.fit(data)
# data_set = pca.transform(data)
```

在训练过程中发现网络出现问题，loss无法进行优化，准确率不上升，在40-50之间。

或许是在降维时对数据造成了损坏，暂时未找到解决方案，或许可以更换别的采样算法，如Gibbs采样进行数据采样，之后的工作来进行实验。

总结

对于机器学习，其对数据的要求是很高的，如果可以将其特征很好的表示出来，可以利用机器学习算法获得较好的效果，但是特征选取的不好会使得结果无法优化。

对于静态的请求来说，在抽取特征时最好考虑的更加全面，结合攻击的实质选取特定的特征，这样会取得好的效果。

另一方面，考虑NLP对其的影响，NLP时对语言的解读，是否能够对请求很好的解读，还是个未知，可以进行研究。

考虑到攻击过程时具有连续性的，所以可以动态的对当前的请求进行分析，结合当前的情景进行判断，或许可以获得更好的效果。

红日AI小组分析工具- 页面展示

利用了flask框架进行了简单架构

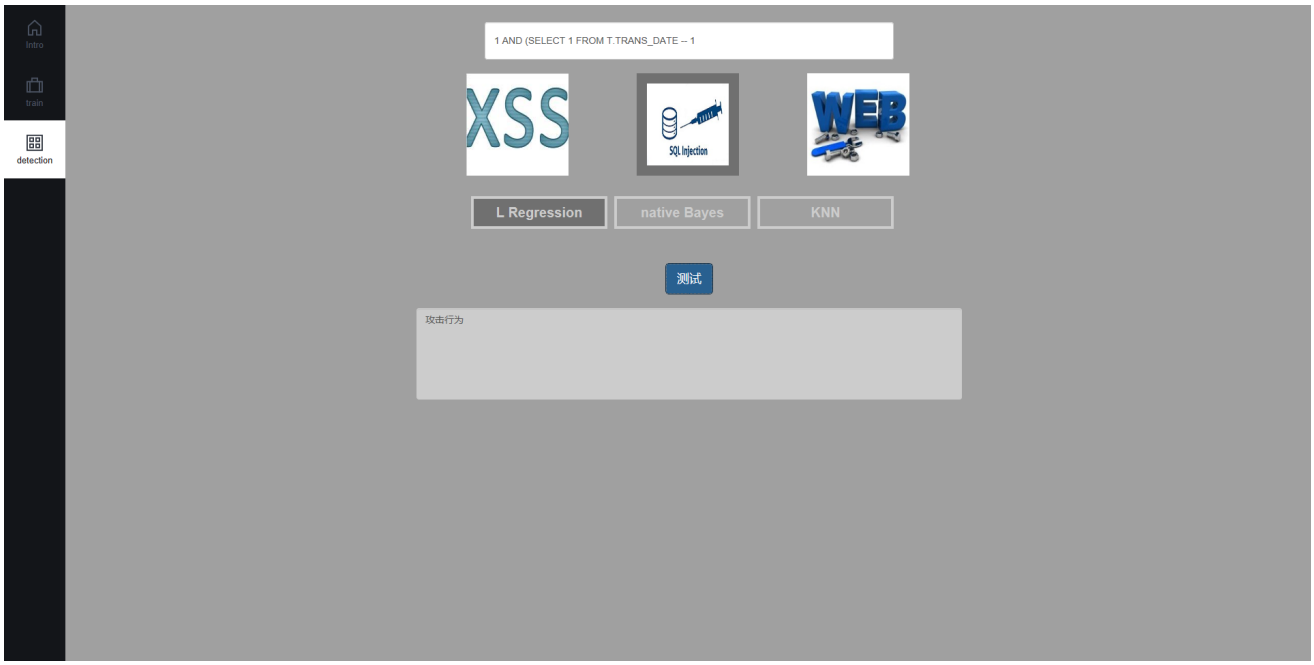
界面分了三个模块，首页、训练页面和检测页面。

首页效果图

训练页面可以自主选择针对的攻击对象，目前是有三种，分别是XSS攻击，SQLI攻击和综合攻击检测；之后进行选择模型使用的算法，包括集中常见的算法，native Bayes、KNN、SVM和LogisticRegression。

上图为选择knn算法进行综合训练的准确率和得分情况

攻击检测页面利用训练得到的模型对输入进行检测，输出检测结果。



上图展示了输入SQL注入攻击串的检测结果

1. 用机器学习玩转恶意URL检测 https://www.freebuf.com/articles/network/131279.html↩