DGA识别

僵尸网络正在威胁着互联网网民的安全。僵尸网络中受到恶意软件感染的僵尸主机由僵尸控制者通过C&C主机进行控制。僵尸主机常常利用DNS授权服务器来解析域名，目的是为了跟C&C服务器创建通信通道，然后获取控制命令，从而进行网络恶意活动。

1 Domain Generation Algorithm

域名生成算法(Domain Generation Algorithm)，是一种利用随机字符来生成C&C域名,从而逃避域名黑名单检测的技术手段。

1.1 运行过程

- 攻击者端

  使用种子运行DGA生成大量域名，随机选择少量的域名进行注册（可能生成了5000个只注册1-2个），攻击者将该域名注册并指向其C&C服务器。

- 受害者端

  使用同样的种子运行DGA，生成大量域名，逐个访问这些域名，检测是否存在，如果该域名未注册，程序继续检测其他域名，如果该域名已注册【如果某生成域名发生了被抢注的情况该怎么办？不处理，因为那些域名并不能发攻击指令，看设计方案是否要继续进行轮询】，那么恶意软件将选择使用该域名联系C&C服务器。

【怎样确定同样的种子？在程序中内嵌。但是安全人员逆向了之后仍旧不能确定种子是什么，因为并不能知道攻击者究竟以什么字段作为种子】

1.2 DGA的危害

DGA每天可以生成成千上万的恶意域名，但仅选择一小部分作为后续的攻击域名，相对于传统硬编码的恶意域名，更难检测。

2 DGA域名识别

2.1 数据集

Alexa是一家专门发布网站世界排名额网站，使用Alexa排名前100万的网站域名作为白样本。

使用360netlab的开放数据作为黑样本。

2.2 特征提取

2.2.1 N-gram

与词袋模型结合，调用CountVectorizer进行转换。

```
vectorizer = CountVectorizer(
    decode_error='ignore',
    ngram_range=(2, 4),
    token_pattern=r'\w',  #按照字符划分
    strip_accents='ascii',
    max_features=max_words,
    stop_words='english',
    max_df=1.0,#作为一个阈值，词是否当作关键词。表示词出现的次数与语料库文档数的百分比
    min_df=1)
```

2.2.2 统计特征模型

```
xxx = [get_aeiou(xx), get_uniq_char_num(xx), get_num(xx), len(xx)]
X.append(xxx)
```

元音字母个数：正常域名通常为“好读”的词，DGA为随机生成，正常域名通常有较多的元音字母。

字母个数：使用set删除重复字母，统计字母个数

数字个数：统计数字个数

当前行文本的长度

2.3 模型训练与验证

2.3.1 gnb

基本原理

条件概率：朴素贝叶斯最核心的部分是贝叶斯法则，而贝叶斯法则的基石是条件概率。贝叶斯法则如下：

$$p(c_i|x,y) = \frac{p(x,y|c_i)p(c_i)}{p(x,y)}$$

这里的C表示类别，输入待判断数据，式子给出要求解的某一类的概率。我们的最终目的是比较各类别的概率值大小。

而上面式子的分母是不变的，因此只要计算分子即可。仍以“坏蛋识别器”为例。我们用C0表示好人，C1表示坏人，现在100个人中有60个好人，则P（C0）=0.6，那么P（**x**，**y**|C0）怎么求呢？注意，这里的（**x**，**y**）是多维的，因为有60个好人，每个人又有“性别”、“笑”、“纹身”等多个特征，这些构成**X**，**y**是标签向量，有60个0和40个1构成。这里我们假设**X**的特征之间是独立的，互相不影响，这就是朴素贝叶斯中“朴素”的由来。在假设特征间独立的假设下，很容易得到P（**x，y**|C0）=P（x0，y0|C0）P（x1，y1|C0）...P（xn，yn|C0）。然而，P（xn，yn|C0），n=0,1,

实现

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
```

2.3.2 XGBoosting

基本原理

二阶展开

实现

```
from xgboost import XGBClassifier
model = XGBClassifier(n_estimate=150, max_depth=9)
```

决策树个数：n_estimate = 100

max_depth	acc
3	0.81
5	0.85
9	0.92

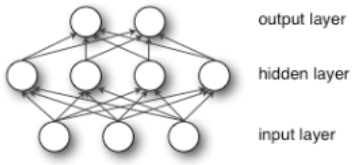
单个决策树最大深度：max_depth = 10

n_estimate	acc
10	0.9154
50	0.9147
150	0.9103

2.3.3 MLP

基本原理

多层感知机（MLP，Multilayer Perceptron）也叫人工神经网络（ANN，Artificial Neural Network），除了输入输出层，它中间可以有多个隐层，最简单的MLP只含一个隐层，即三层的结构，如下图：



隐藏层与输入层是全连接的，假设输入层用向量X表示，则隐藏层的输出就是

f(W1X+b1)，W1是权重（也叫连接系数），b1是偏置，函数f 可以是常用的sigmoid函数或者tanh函数。

实现

```
from sklearn.neural_network import MLPClassifier
model = MLPClassifier(hidden_layer_sizes=(5, 2), activation='relu', solver='adam',
shuffle=True, verbose=1)
```

终止：Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.

2.3.4 LSTM

基本原理

实现

```
#输入尺寸为（n_sampels,max_words)
input_dim = max_words
max_features = 100
model = keras.models.Sequential(
    [keras.layers.Embedding(input_dim=max_features+1, output_dim=128,
input_length=input_dim),
     keras.layers.LSTM(units=128, dropout=0.2, return_sequences=True),
keras.layers.LSTM(units=128, dropout=0.2),
     keras.layers.Dense(units=128, activation="relu"),     keras.layers.Dropout(0.2),

     keras.layers.Dense(units=1, activation="sigmoid")])
model.summary()
model.compile(loss="binary_crossentropy", optimizer=keras.optimizers.Adam(), metrics=
['accuracy'])
```

需要固定输入尺寸，先对数据进行转换或填充。