Python Programs

Arpit Gawande

February 14, 2018

1 Data Extraction from Wireshark capture file

```
# Common Imports
#Pandas for creating dataframes
import pandas as pd
#Pyshark to capture packets
import pyshark
#For file operarions
import os

# Reading packets from pre-captured file
file_cap = pyshark.FileCapture('captures/latest/attack2.pcapng')

# Filter data from the captured packets for further processing

#Attributes for sample size
startTime = 0.0
endTime = 0.0
i = -1
first = True
#Write extraction result to
base_directory = 'converted/test2/attack_samples/2/'
#Sample Number
init_sample = 1
dfList = []
while(True):
    i += 1
    #Each packet can have multiple layers (e.g eth, ip, tcp etc.).
    #Combine them in list for a single packet.
    layerList = []
    #pyshark not able to handle AttributeError, AssertionError and KeyError(index values)
    try:
        #Iterate through layers of packet
        for layer in file_cap[i]:
          # Slice Data according to time
          t = float(file_cap[i].sniff_timestamp)
          #Initial setup
          if startTime == 0.0:
              startTime = t
              endTime = startTime + 300 # 300 sec(5 min) is the sample size
              sample = init_sample
              print(sample, startTime, endTime)
          # Write every sample to csv file
          elif t > endTime:
              if not os.path.exists(base_directory):
                  os.makedirs(base_directory)
              dfList.to_csv(base_directory+str(sample))
              #Clear list after writing this would save memory
              startTime = endTime
              endTime = startTime + 300
              sample += 1
              #Reset after writing to file
              first = True
              print(sample, startTime, endTime)
              gc.collect()
          #We need only ip layer only
          if layer._layer_name == 'ip':
              #Layer values are in the form of dictionary. Filter the attributes
```

```
                layer_dict =
                {key:value for key, value in layer._all_fields.items() if key in required_keys}
                #Create dataframe from dictionary
                layerList.append(pd.DataFrame(layer_dict, index=[0]))
                #Add timestamp and sample number
                layerList.append(pd.DataFrame({'sniff_timestamp':file_cap[i].sniff_timestamp}, index=[0]))
                layerList.append(pd.DataFrame({'sample':sample}, index=[0]))
                #Build packet dataframe from layer frames. Its single row dataframe
                cDf = pd.concat(layerList, axis=1);
                if first:
                    dfList = cDf
                    first = False
                else:
                    dfList = dfList.append(cDf, ignore_index=True)
    except (AttributeError, AssertionError) as e:
            continue  #print('Ipv4 packet does not exist')
    except  KeyError:
            break;

#If sample data is not written to file because for frame size
if(sample == init_sample):
    dfList.to_csv(base_directory+str(init_sample))
```

2 Clustering based on destination IP address

```
# Common Imports
#Pandas for creating dataframes
import pandas as pd
#Sklearn
from sklearn import preprocessing
#K-means clustering algo
from sklearn.cluster import KMeans
#OS moduled for file oprations
import os
#CSV module
import csv

#Folder Base Path
base_path = 'converted/test2/'

#Cluster Path
#Normal
sample_path = base_path+'samples/'
cluster_path = base_path+'ip_cluster/'
#Attack
#sample_path = base_path+'attack_samples/2/'
#cluster_path = base_path+'attack_ip_cluster/2/'

first = True
ip_dict = dict()
sample_count = 1;
for filename in os.listdir(sample_path):
    tdf = pd.read_csv(sample_path+filename, index_col=0)
    #Filter Columns
    t = tdf[['ip.dst', 'ip.proto', 'sniff_timestamp', 'sample']]
    #Remove null destinations
    t = t[t['ip.dst'].notnull()]
    #Rename Columns
    t.columns = ['ip', 'proto', 'time_stamp', 'sample']
    #Get count for each ip
    df = t.groupby(['ip', 'proto']).size().unstack().fillna(0).astype(int)
    #Select TCP and UDP as only fetures (TCP:6, UDP:17)
    #df = df[[6,17]]
    #Get value matrix
    X = df.values
    #Create scaling
    scaler = preprocessing.StandardScaler().fit(X)
    #Transform Traning data
    X_trans = scaler.transform(X)
    #Define Number of Clusters
    cluster_count = 6
    #Data Fitting using K-means
    if first:
        kmeans = KMeans(n_clusters=cluster_count)
        kmeans.fit(X_trans)
        centroids_array=kmeans.cluster_centers_
    else:
        kmeans = KMeans(n_clusters=cluster_count, init=centroids_array)
        kmeans.fit(X_trans)
    #Attaching label/cluster to IP
    ip_series = pd.Series(df.index, name='ip')
```

```
    label_series = pd.Series(kmeans.labels_, name='label')


    ip_label_df = pd.concat([ip_series, label_series], axis=1).set_index('ip')
    if not os.path.exists(cluster_path):
        os.makedirs(cluster_path)
    ip_label_df.to_csv(cluster_path+str(sample_count))
    sample_count += 1

#Cluster Assignment
from itertools import groupby
ip_dict = dict()

for filename in os.listdir(cluster_path):
    with open(cluster_path+filename, newline='') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            if row['ip'] in ip_dict:
                #print(row['ip'],ip_dict[row['ip']])
                    ip_dict[row['ip']] = ip_dict[row['ip']] + [row['label']]
                else:
                    ip_dict[row['ip']] = [row['label']]

#Find how many time destination has been assigned a given cluster
ip_cluster_dict = dict()
for key, value in ip_dict.items():
    ip_cluster_dict[key] = {k: len(list(group)) for k, group in groupby(value)}
```

3 SVM learning for IP address with highest traffic

```
# Common Imports
#Pandas for creating dataframes
import pandas as pd
#Sklearn
from sklearn import preprocessing
#OS moduled for file oprations
import os
#CSV module
import csv
#SKlearn SVM
from sklearn import svm

#Folder Base Path
base_path = 'converted/test2/'

first = True

ip_dict = dict()
sample_path = base_path+'samples/'
#Cluster Path
#cluster_path = base_path+'attack_ip_cluster/2/'
sample_count = 1;
first = True;
dfList = []
for filename in os.listdir(sample_path):
    tdf = pd.read_csv(sample_path+filename, index_col=0)
    #Filter Columns
    tdf = tdf[['ip.dst', 'ip.proto', 'sniff_timestamp', 'sample']]
    #Remove null destinations
    tdf = tdf[tdf['ip.dst'].notnull()]
    #Rename Columns
    tdf.columns = ['ip', 'proto', 'time_stamp', 'sample']
    #Get count for each ip
    df = tdf.groupby(['ip', 'proto']).size().unstack().fillna(0).astype(int)
    df['sample'] = filename
    dfList.append(df)

#Group all the flow for an IP address for SVM training algorithm.
df1 = pd.concat(dfList).sort_index()
df1 = df1.reset_index().set_index(['ip','sample'])
indexValues = df1.index.get_level_values(0)

#Use SVM to train on every destination IP
svm_dict = dict()
for i in indexValues:
    X_train = df1.loc[i].values
    # fit the model
    clf = svm.OneClassSVM(nu=0.1, kernel="rbf", gamma=0.1)
    clf.
    clf.fit(X_train)
    svm_dict[i] = clf

#Predict for the give destination if it is normal or not
svm_dict['192.168.0.7'].predict([[0,0,1,1]])
```