Project

Project Title: Understand How VPN Traffic was Analyzed
Group Authors: Hsin-ih Tu

Dataset

www.unb.ca/cic/datasets/vpn.html

Problem/Intro

How did the researchers detect categories of activity in VPN network traffic in the Draper-Gil, Lashkari, Mamun, and Ghorbani paper [1]? My understanding of VPN is virtual network is created that is insulated from the outside so it made me wonder what techniques did the researchers use because I was under the impression that VPN traffic is not exposed. Can we reproduce the results from the paper and be able to identify the 7 categories of traffic (web browser, email, chat, streaming, file transfer, voIP, P2P)? by attempting to train two network traffic classifiers.

Previous work

The reference paper describes related work done by other teams that are dependent on the payload, or data packet. So most of the analysis relies on searching for patterns with the information that is transmitted. This can be less effective when connections use encryption. Another weakness described is the related works have been narrower in scope by concentrating on specific traffic or specific devices. The paper proposes to analyze network traffic using time related features, and across more than one set of traffic category.

Approach

We only follow the methods from the paper that involve the VPN subset. This is because my interest centered much more on the fear that VPN traffic was vulnerable to detection. The algorithms specified were C4.5 and KNN. Cross validation with 10 folds are also used according to the paper. For the training and testing data, Standard scaling and PCA is applied in order to prevent axis [3] skew.

Experiments

The first step was to read the dataset into a dataframe. The archive files hosted at the UNBCA URL are organized into two subdirectories: raw PCAP network capture data, and delimited data (CSV) with the .arff file extension. Without knowing what the difference was, I avoided the PCAP files as their size exceeded 20 GB. Once unzipped the CSV data is separated into three scenario subdirectories: A1, A2, B. Within the scenarios there are files collecting the time related data into timeouts of 15, 30, 60, 120. Starting with the A2 non-VPN 120-timeout data, examining the info (head, describe) and sample rows the only thing we can determine is columns with names such as "Unnamed NN". It seems that column 23 contains the human readable label of the traffic designation. Looking at the value_count of column 23 seems to confirm. The additional note is that the first 26 rows seem to be header related, and mostly contain NaN which is the reason we skip them in preparing the training and testing subsets. For the dataset split, we used the value 0.2 for test subset.

For the Decision Tree algorithm, the SciKit Learn library provides `DecisionTreeClassifier`. We create the model and choose a `max_depth` parameter value of half the number of columns; this maximum depth is a guess instead of doing multiple trial/error tests. This is one way to regularize and reduce overfitting in the decision tree.

Next, we call the fit method to train on the training subset. For an embarrassing long time, I struggled to format the training feature data and training label data into the proper types expected by the fit method. With the feature training data, it can actually be passed straight from the output result of the `train_test_split` call. However, we follow the steps described in the O'Reilly book to combine Standard scaling with PCA into a pipeline. The output result of this pipeline becomes the feature training subset that is passed as the first parameter of the fit call.

For the second parameter of the fit call, we transform the training label data. The `OneHotEncoder` will convert the text into the sparse matrix of ones and zeros. So we create the encoder, but to call its fit method the training labels need to be two dimensional in shape. This can be done with the `reshape` method. The new shape we want means "many observations of one feature" which is a way of saying these category labels are different values of one enum. We specify the reshape parameters (-1, 1) to mean "guess the n-many rows, and one column". This two dimensional shape is ready to pass to the encoder's fit method. Then after the fit is done, the encoder can be used to call transform on more two dimensional shaped training labels. Which we do and call the transform and pass the same two dimensional shaped training labels to produce the encoded list. This encoded list is almost ready to be used for the DecisionTreeClassifier's fit method. It just has to be in array format. So for convenience, we call the to-array method and store that in the `labels_train` variable. This variable can be finally passed as the second parameter of the DecisionTreeClassifier's fit method.

To evaluate the model, we shape the test subset into parameters for the cross validation to make prediction results. Then we take these prediction results as input parameters to the classification report.

For the K-Nearest Neighbor algorithm, the Scikit Learn library provides `KNeighborsClassifier`. We create the model and call the fit method to train on the training subset. We call the cross validation with the training subset for the prediction values that will be input to the average scores for F1, Precision, and Recall. Finally, we call the cross validation with the testing subset for the prediction values that will be input to the classification report.

One value that can be a factor in the KNN classification report comes from the handling of unpredicted labels. The call returns a warning about divide-by-zero, and mentions a `zero_division` parameter. The default behavior is to use a zero value in the calculation and return the warning. For our experiments, we assign the `zero_division` parameter as `np.nan` to signify not-a-number.

We repeated these steps for the different VPN dataset files of timeouts 15, 30, 60.

Conclusions

Using the Draper-Gil, Lashkari, Mamun, and Ghorbani paper [1] as reference we wanted to learn how VPN network traffic can be grouped into categories. We used the DecisionTreeClassifier and KNeighborsClassifier to approximate the research procedures that involved C4.5 and KNN. We used the dataset provided for scenario A2 (VPN 15/30/60 timeout). The differences observed can be caused by any combination of the algorithm default parameters, split size of train/testing sets, and the random seed value. The important understanding that I gained was that classification achievable with machine learning does not mean that the content of VPN traffic is revealed or decrypted. In other words, my concern was that the research had found methods to defeat the primary purpose of VPN (common example of remote employees need for secure connections to the office network).

As future work, I need to study the CICFlowMeter tool [2] created by the researchers. It was named ISCXFlowMeter in its first incarnation. Since then, CICFlowMeter reached a 4th version. The community has also ported the tool (from Java) to Python, and more recently C++. Based on the summary of the extraction of the time related features, the definition of a flow is: packets that have the same Source-IP, Destination-IP, Source-Port, Destination-Port, Protocol (TCP/UDP) [1]. How flexible is this definition? There is also the question of whether it is possible for a VPN server to evade the classifier by mimicing the categories that performed the worst (i.e., category #1).

```
timeout-15 DTC classification report:
              precision    recall  f1-score   support

           0       0.73      0.75      0.74       513
           1       0.46      0.42      0.44       233
           2       0.73      0.66      0.69       411
           3       0.60      0.74      0.66        95
           4       0.66      0.74      0.70       170
           5       0.79      0.81      0.80       108
           6       0.96      0.97      0.96       429

    accuracy                           0.74      1959
   macro avg       0.71      0.73      0.71      1959
weighted avg       0.74      0.74      0.74      1959

KNN F1: 0.761450984391107
KNN precision: 0.8087443696159463
KNN recall: 0.728034889235558
KNN report:
               precision    recall  f1-score   support

           0       0.77      0.82      0.79       513
           1       0.65      0.41      0.50       233
           2       0.78      0.60      0.68       411
           3       0.68      0.66      0.67        95
           4       0.72      0.70      0.71       170
           5       0.78      0.78      0.78       108
           6       0.98      0.97      0.98       429

   micro avg       0.80      0.74      0.77      1959
   macro avg       0.77      0.71      0.73      1959
weighted avg       0.79      0.74      0.76      1959
 samples avg       0.80      0.74      0.74      1959


timeout-30 DTC classification report:
              precision    recall  f1-score   support

           0       0.77      0.85      0.81       528
           1       0.51      0.44      0.47       149
           2       0.61      0.51      0.56       205
           3       0.82      0.82      0.82       175
           4       0.75      0.74      0.75       175
           5       0.79      0.78      0.79        59
           6       0.96      0.96      0.96       256

    accuracy                           0.77      1547
   macro avg       0.74      0.73      0.74      1547
weighted avg       0.76      0.77      0.76      1547

KNN F1: 0.7990925067241735
```

KNN precision: 0.8286870416403358
KNN recall: 0.7768344841459086
KNN report:

	precision	recall	f1-score	support
0	0.78	0.85	0.81	528
1	0.55	0.32	0.41	149
2	0.70	0.40	0.51	205
3	0.79	0.78	0.78	175
4	0.76	0.75	0.76	175
5	0.77	0.69	0.73	59
6	0.98	0.96	0.97	256
micro avg	0.79	0.73	0.76	1547
macro avg	0.76	0.68	0.71	1547
weighted avg	0.78	0.73	0.75	1547
samples avg	0.79	0.73	0.73	1547

timeout-60 DTC classification report:

	precision	recall	f1-score	support
0	0.77	0.82	0.80	513
1	0.33	0.28	0.30	104
2	0.53	0.45	0.49	190
3	0.68	0.65	0.66	71
4	0.73	0.74	0.73	151
5	0.56	0.65	0.60	34
6	0.87	0.90	0.88	324
accuracy			0.73	1387
macro avg	0.64	0.64	0.64	1387
weighted avg	0.72	0.73	0.72	1387

KNN F1: 0.745737344932684
KNN precision: 0.787943180025375
KNN recall: 0.7143672834966244
KNN report:

	precision	recall	f1-score	support
0	0.77	0.83	0.80	513
1	0.63	0.26	0.37	104
2	0.74	0.37	0.50	190
3	0.67	0.55	0.60	71
4	0.69	0.66	0.68	151
5	0.76	0.56	0.64	34
6	0.89	0.86	0.87	324
micro avg	0.78	0.69	0.73	1387
macro avg	0.74	0.59	0.64	1387
weighted avg	0.77	0.69	0.72	1387
samples avg	0.78	0.69	0.69	1387

References

1. Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, Ali A. Ghorbani, "Characterization of Encrypted and VPN Traffic Using Time-Related Features", In Proceedings of the 2nd International Conference on Information Systems Security and Privacy(ICISSP 2016) , pages 407-414, Rome, Italy.
2. CICFlowMeter network traffic flow generator, https://www.unb.ca/cic/research/applications.html; (source repo https://github.com/ahlashkari/CICFlowMeter).
3. Aurélien Géron, "Hands-on machine learning with Scikit-Learn, Keras and TensorFlow", 2023, O'Reilly Media, Inc.

Source repo

https://github.com/patterns/eep567/