A Report on Packet Sniffer

CS 375 Project Report

KENTON CARRIER, CHELINA ORTIZ MONTANEZ, and JONATHAN MORFORD

Abstract

By capturing the traffic traversing a network and analysing the features of the packets and connections between hosts, the network-based application in use can be predicted utilizing machine learning technologies. Through our analysis, we found that Support Machine Vector algorithms were able to best identify the correct type of network activity based on our training set.

ACM Reference Format:

Kenton Carrier, Chelina Ortiz Montanez, and Jonathan Morford. 2019. A Report on Packet Sniffer: CS 375 Project Report. 1, 1 (April 2019), 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A packet sniffer is one form of data gathering that relays that information to its creator. This piece of software can scan any level of information, from the Ethernet characteristics, specific details about the IP protocol, size of the packets sent, among other data regarding the network traffic. In our project, we aim to understand this type of software and explore ways to use it without being intrusive to the network, as well as to understand how

Authors' address: Kenton Carrier; Chelina Ortiz Montanez; Jonathan Morford.

machine learning can predict and make assumptions of what the machine is doing on the Internet.

2 PROPOSED FEATURES

We chose flow features that we thought would maximize the classification strength of the machine learning algorithms by finding features that varied between each type of activity we tracked. The simplest features are the IP protocol and the number of packets in the flow. Each packet in a single flow used the same protocol, either TCP or UDP and we saw that certain activities used one or the other more frequently. The number of packets involved in the flow also helped differentiate between activities since some activities involve larger amounts of data transfer in a single connection. Our other features involved more complex calculations, and we decided to analyze the differences in the amount of data being sent and received in each flow. Since we detected our flows bidirectionally, we wanted to see the differences in the percentage of packets flowing into our device and the amount flowing out. In the case of video streaming and file downloading, these percentages should favor packets coming in, but video conferencing should have a more even split of incoming and outgoing packets. We also decided to look at the maximum sizes of the packets coming in and out as well; during our initially testing and analysis we saw consistency among the maximum size of the packets coming into the device and out of the device during certain activities.

3 IMPLEMENTATION

For the first part of the project, we used the provided skeleton as a base for sniffing the packets. After sniffing the packets, we loop through the packets and create flows based on the protocol used, source and destination IP, and source and destination ports. If a packet does not match one of the flows we have already detected, we create a new flow from that packet's information. If the packet does match the path in either direction from one of our previously detected flows, it is added to that flow and the flow's features are recalculated with the new packet's information.

Our features are calculated as packets are identified as being a part of a flow. The number of packets is simply incremented each time a new packet is added to the flow. For the maximum size of the packets coming in and going out, we do a simple check if the size of the packet is larger than the current max in the direction that the new packet is detected. For example, if a new packet is received, we check if the size of the packet exceeds the maximum size of packets coming in, and if it does, we replace the size with the size of the new packet. The most complex is the percentage of packets incoming and outgoing. This is done by taking the current percentages and adding the new packet to the number of packets moving in its direction. Then, we recalculate the percentage of packets moving in that direction and update the opposite statistic by subtracting the new percentage from 1.

For the machine learning portion of the project, we created a diverse training set for our model. The focus for the training set was to be thorough and precise in adding flows to make our prediction as accurate as possible. To create this training set, we isolated the desired activity on our device and ran the packet sniffing script; this output the detected flows to a csv that was then checked for accuracy and appended to our training set. This process was repeated until we had 25 data points for each activity. In order to provide predictions for a variety of applications, we added flows from several different streaming services, different video conferencing apps, multiple files from different websites, and browsing data from varying websites. Then, we applied 3 types of machine learning algorithms to our sets: decision trees, neural networks and SVM. Each was configured to use training x and y coordinates and from there predict the accuracy of the flow calculations in future experiments.

4 EXPERIMENTS

We used three different algorithms – Decision Trees, Support Vector Machines, and Neural Networks – to determine which one could most accurately predict what kind of activity that the hosts engage in. The training data includes flows from each of our features to ensure that we can easily identify four main scenarios on the network: web browsing, video streaming, video conferencing and file downloading.

A Decision Tree is a method used to teach the machine to select the best attribute from the training set, make it a decision node and create smaller subsets from that node, so that after doing this process continually until there are no attributes left. The partition of data

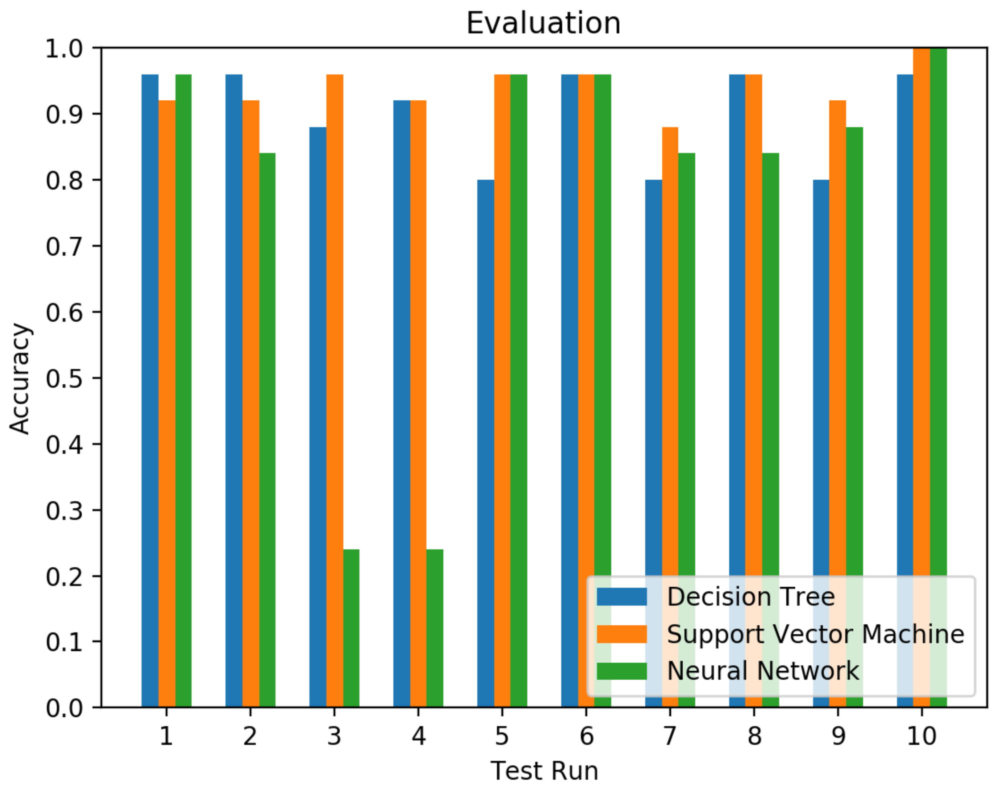

Fig. 1. Comparison of each machine learning algorithm used for testing our training set.

into subsets is done through recursion and it is more efficient in terms of time complexity so it deals with great amounts of data accurately.

A Support Vector Machine (SVM) is a type of classifier that discriminates based on a separating hyperplane. Using the training sets we input, the algorithm defines the hyperplane that fits the data into specific categories. Using a linear hyperplane has yielded the most accurate predictions for our dataset.

A Neural Network is an algorithm that predicts the categories that the data input belong to. It uses perceptrons, which are linear classifiers that separate the inputs into two categories, depending on their weight, and separates them linearly. Multilayer Perceptrons (MLP) is an artificial deep learning type of a Neural Network. It is divided into layers that receive a signal, make a prediction about the data provided and do other hidden processes that determine the correlation of each to a specific category.

To gauge the accuracy of these algorithms, we had the software divide up the training data into a 75-25 partition, where 75% of the data was used to train the machine learning models, and 25% of the data was used to test the trained models afterwards. These divisions are randomly determined for each test run iteration, allowing the accuracy of the model to be tested for different training sets. Overall, the SVM algorithm was best able to fit our training data consistently, yielding an average accuracy of about 95% over 4000+ test runs. Decision Trees had a very similar, yet consistently lower accuracy of about 93%. Further testing would be required to determine if this drop in accuracy is specific to this dataset or to the algorithm. Neural Networks did not perform consistently at all between iterations as can be seen in the figure below, and only had an average accuracy peaking just below 75%.

5 CONCLUSIONS

After developing this packet sniffer, we have found that network traffic follows patterns among the four activities we analyzed. While our tool is meant for educational purposes and would be useful in a testing environment, these kinds of applications can be exploited to extract private information from users. This analysis shows how important creating reliable and secure networks and networking protocols are in protecting private data.

6 REFERENCES

- NAVLANI, A. 2018. Decision Tree Classification in Python. DataCamp. https://www.datacamp.com/community/tutorials/decision-tree-classification-python
- SKYMIND. N. D. A Beginner's Guide to Multilayer Perceptrons (MLP). https://skymind.ai/wiki/multilayer-perceptron
- PATEL, S. 2017. SVM (Support Vector Machine) - Theory. Machine Learning 101, Medium. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72