# DS 6999: Research Project

# Detecting and Analyzing Malicious IP Addresses in Regular Traffic

# Murugesan Ramakrishnan, mr6rx@virginia.edu

**Objective:**

The objective of the research was to analyze the regular traffic and identity potential malicious IP addresses. This could be achieved with the help of honeypot data, which contains a list of malicious IP addresses obtained from honeypot servers.

**Project Overview:**

The analysis consisted of two sets of data – conn logs and the honeypot data. The conn logs contained regular network traffic and the honeypot data had potential malicious IP addressed to be flagged. While conn logs had pre-defined parsers to read the data in, I developed parsers to read the honeypot files and would return *pandas data frame,* easier for further processes.

Post reading in the normal and honeypot traffic in Python, malicious IPs in normal traffic had to be identified based on honeypot logs. Since the IP addresses change over time I had to lookup not just the malicious IP addresses but also their corresponding timestamp information. I developed a search algorithm that would take in a time window and search for all the malicious IP addresses in regular traffic that match with honeypot IPs within that specified time window. The output dataset would contain a flag *honeypot\_flag* to indicate the malicious traffic.

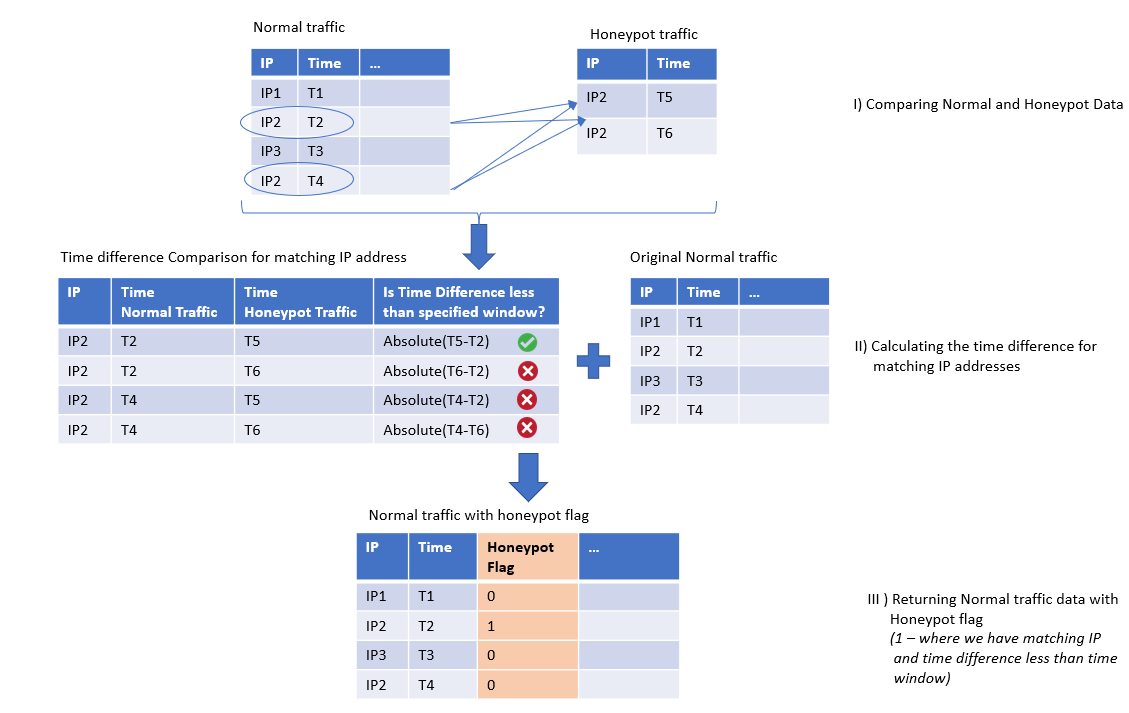
Flag indicator was used from the data received from the search algorithm function to segregate the data into malicious and regular traffic which helped in creating descriptive plots to understand primary differences between the two. I created modular plot functions which can be conveniently called after executing the search algorithm.

**Reading Data:**

Regular traffic data (conn logs) were obtained from bro logs, which contained flow level data (aggregated version of pcap data which contains aggregate level statistics about the packets exchanged between a source and destination over a certain time interval). Since the bro log parsers are readily available “ParseBroLogs” library was used to read in the bro logs. However, for honeypot data since there are no parsers readily available, python codes were written to identify each column headers and its contents using regular expressions thereby precisely reading each line and creating a data frame. It was made sure that the parsing code is scalable to accommodate any number of input fields. During the course of research, we had an alternate data source called Fireeye to get the malicious IP addresses (proxy for honeypot data) for which the data was not in a structured format. Parsing functions had to be written to segregate various fields using string functions. But since the data did not have proper source and destination IP information we proceeded with honeypot data despite not getting a common day data between honeypot and conn logs

**Implementing Search Algorithm:**

After reading in the data, I compared the two files to identify the fields required for the algorithm. Two main columns were required - IP address details and timestamp. To keep the timestamp consistent, unix time format (as present for regular traffic) was used. The time format in honeypot was converted to unix format to match the regular traffic data. A function *search\_algo* was defined to get in the two datasets, time window and the names of ip address and time stamp columns. The function first does a left join on the normal traffic (conn logs) using honey pot data. After the join, timestamp difference for the matched records are calculated. If the difference is lesser than the specified time window, those IP addresses are noted. Now, a new flag column to the normal traffic data is created which is 1 for the noted malicious IP address and 0 for the rest.



*General Workflow of the Algorithm*

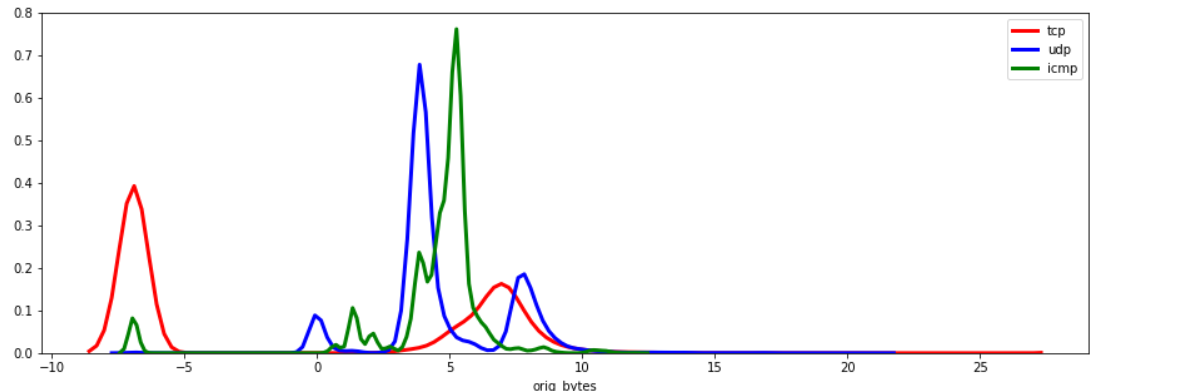
**Efficiently analyzing the results:**

Once the malicious IPs were flagged in the regular traffic, various analysis were performed. The plot analysis was modularized by creating a class *plot\_analysis*, which takes in the resulting data from search algorithm and provides intuitional results. Following are the different analysis generated by the class,

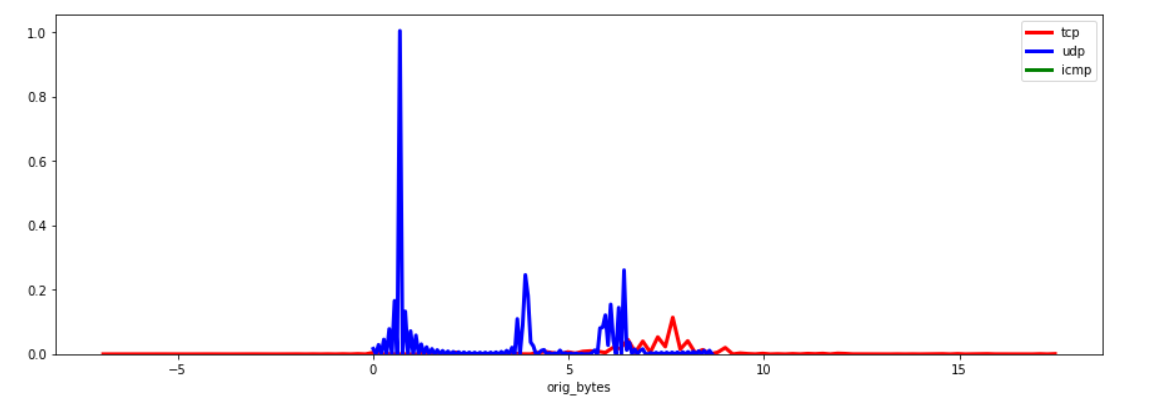
1. *Frequency of attack:* How often do we get an attack from a malicious user (hourly, daily and monthly levels)
2. *Top n IP address and Ports:* What are the top IP addresses and ports from which we get an attack.
3. *Proportions and distribution of protocols:* What proportion of tcp, udp and icmp do we have for normal and malicious traffic.
4. *Distribution of network connection duration:* What is the distribution of duration for the two traffic types.

Following are some of the plots generated on imputed data,

1. *Distribution of protocols in log scale for normal traffic*

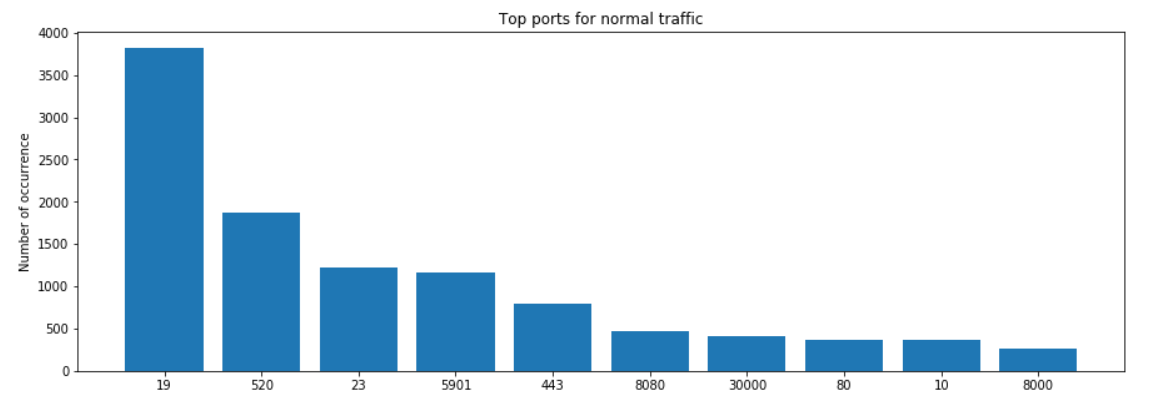


1. *Distribution of protocols in log scale for malicious traffic*



The density for malicious traffic is lesser because of less amount of data.

1. *Top 10 ports for malicious traffic:*



**Result:**

The search algorithm was successfully implemented and tested with imputed data. Exhaustive testing can be done once new set of data is received from the honeypot servers. With ready-to-execute plot modules, we can understand the major differences between the two traffics due to variables like orig\_bytes, duration, ports, IP address, protocols. With this analysis, it would be easier to run a classification algorithm with honeypot\_flag as the dependent variable. This will help explain the feature importance.