**TEAM DETAILS**

**Team Name –** Ctrl Intelligence **Team Members** – Shrey Bansal, Kalash Gupta, Mukul Singh

# BOTNET DETECTION FROM NETWORK FLOW ANALYSIS

## What is Botnet?

Botnet is one of the most dangerous cyber-security issues. A botnet infects unprotected machines and keeps track of the communication with the command and control server to send and receive malicious commands. 'e attacker uses botnet to initiate dangerous attacks such as DDoS, fishing, data stealing, and spamming. 'e size of the botnet is usually very large, and millions of infected hosts may belong to it.

Studies stated in the reference section have shown that botnets have a unique communication and conversation pattern that can be used for identifying botnet activities. We have used this approach of analysing the communications of a host to classify it as botnet or benign.

## Proposed Solution

The solution is based on extracting network traffic flows from packet capture data of a network. Features are then extracted from each flow after appropriate selection and engineering. These flows are then clustered based on the extracted features. Clustering is used to remove the flows that are surely benign. The remaining flows are sent to a classifier for final prediction. After the flows have been classified, each host is scored based on the percentage of malicious flows originating from it. The hosts are then flagged based on the percentage of their connections labelled as malicious. We have basically used a **combination of supervised and unsupervised learning techniques**. The solution proposed has 7 steps namely,

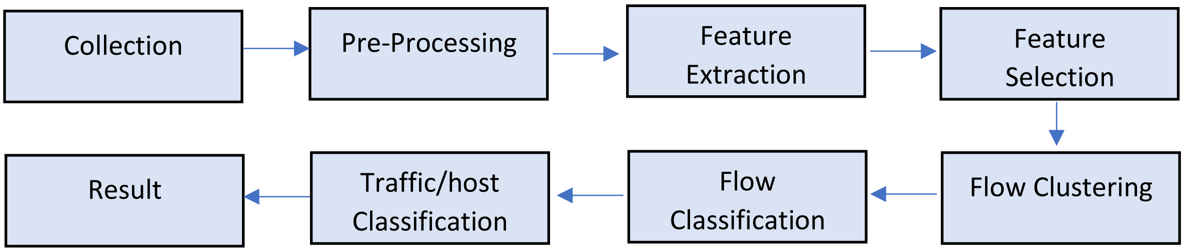
1. **Data Collection**
2. **Data Pre-Processing**
3. **Feature Extraction**
4. **Feature Selection**
5. **Clustering of flows**
6. **Flow Classification**
7. **Traffic Identification**

**Data Labelling –**

We have used traffic flows for our classification. The flows originating from the host IP have been given the label of the host, i.e. for data captured on a Benign system, only the flows originating from the benign system are labelled benign, rest flows are not considered/ignored.

Each of the step has been briefly described in the following section.

**Solution Architecture**



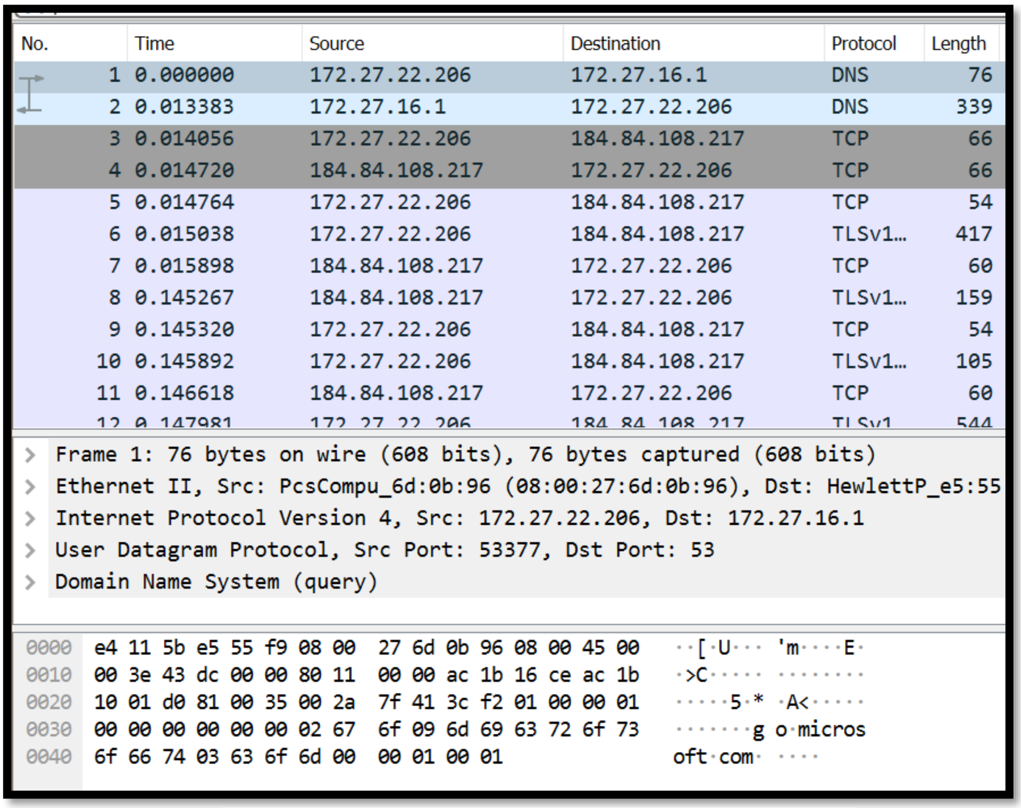
### ➔ Data Collection –

PCAP file or a network capture file contains packet traffic data of a host on a network. The nature of the host is known as either botnet or benign.

**Files in the Storm folder were not Ethernet packages but rather Cooked Linux Frames. A direct PCAP filter would have ignored these files. To overcome this, the CookedLinux Layer of Scapy was used. This ensured no packet data is lost and the model trains on the entire dataset.**

Network flow is a series of packets exchanged between two distinct hosts. A network flow is identified by source and destination ports and IP along with the protocol used. For a benign host, all flows originating from that host are labelled benign. That is for every flow which has source IP as the host are labelled based on the host nature.

Since PCAP and PCAPNG files comprise of memory blocks and cannot be manipulated directly. Wireshark along with the command line tool TShark were used for the manual analysis of the data. A sample of a Wireshark analysed PCAP file is shown below.



| No. | Time | Source | Destination | Protocol | Length |
|---|---|---|---|---|---|
| 1 | 0.000000 | 172.27.22.206 | 172.27.16.1 | DNS | 76 |
| 2 | 0.013383 | 172.27.16.1 | 172.27.22.206 | DNS | 339 |
| 3 | 0.014056 | 172.27.22.206 | 184.84.108.217 | TCP | 66 |
| 4 | 0.014720 | 184.84.108.217 | 172.27.22.206 | TCP | 66 |
| 5 | 0.014764 | 172.27.22.206 | 184.84.108.217 | TCP | 54 |
| 6 | 0.015038 | 172.27.22.206 | 184.84.108.217 | TLSv1… | 417 |
| 7 | 0.015898 | 184.84.108.217 | 172.27.22.206 | TCP | 60 |
| 8 | 0.145267 | 184.84.108.217 | 172.27.22.206 | TLSv1… | 159 |
| 9 | 0.145320 | 172.27.22.206 | 184.84.108.217 | TCP | 54 |
| 10 | 0.145892 | 172.27.22.206 | 184.84.108.217 | TLSv1… | 105 |
| 11 | 0.146618 | 184.84.108.217 | 172.27.22.206 | TCP | 60 |
| 12 | 0.147981 | 172.27.22.206 | 184.84.108.217 | TLSv1 | 544 |

```
> Frame 1: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
> Ethernet II, Src: PcsCompu_6d:0b:96 (08:00:27:6d:0b:96), Dst: HewlettP_e5:55
> Internet Protocol Version 4, Src: 172.27.22.206, Dst: 172.27.16.1
> User Datagram Protocol, Src Port: 53377, Dst Port: 53
> Domain Name System (query)
```

```
0000  e4 11 5b e5 55 f9 08 00  27 6d 0b 96 08 00 45 00   ··[·U··· 'm····E·
0010  00 3e 43 dc 00 00 80 11  00 00 ac 1b 16 ce ac 1b   ·>C····· ········
0020  10 01 d0 81 00 35 00 2a  7f 41 3c f2 01 00 00 01   ·····5·* ·A<·····
0030  00 00 00 00 00 00 02 67  6f 09 6d 69 63 72 6f 73   ·······g o·micros
0040  6f 66 74 03 63 6f 6d 00  00 01 00 01               oft·com· ····
```

**Wire Shark Analysis Window**

## ➔ Data Pre-Processing

Since a PCAP file only contains raw samples and they need to be aggregated to have labelled data. In our solution we decided to extract flows from the files as botnet activities detection based on flows have shown success in referenced works.

Each flow was characterized by a 5-tuple of (Source IP, Destination IP, Source Port, Destination Port, Protocol). All packets are aggregated to belong to one flow. This gave us about 80-90 thousand flows per subdirectory. For all the subdirectories, every pcap file was merged with timestamp to get a single subdirectory.csv file (containing all the information for flows).

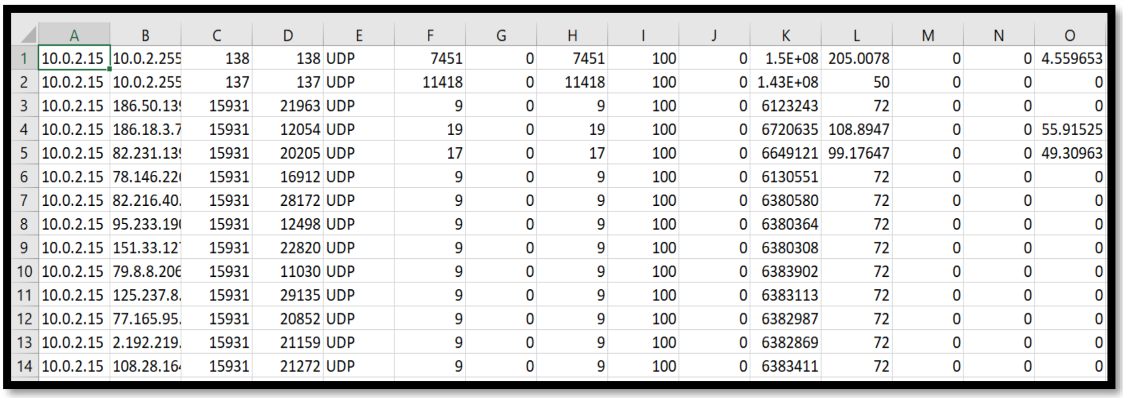
Since we only know about the nature of the host who captures the data and not of individual flows, we need a labelling system. For this, all flows that originate from the known IP are given the label of the IP, i.e. benign for a benign IP. This eliminated about 50% of the flows.

**All features have been scaled using Min-Max scaling technique to improve model performance**

The remaining flows still were too large to be processed efficiently. Since, we are only concerned with P2P activities, flows with source port lower than 1000 were removed as these usually correspond to normal non-P2P internet traffic.

Flows containing less than or equal to 2 packets were also removed since they would not provide much data on the connection nature but will only skew the model. Also 1 packets flow are usually only handshake requests on not P2P.

The flows filtered by the above techniques were considered for model training for botnet detection. A csv file of the flows was generated. A screenshot of the generated file is shown below.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10.0.2.15 | 10.0.2.255 | 138 | 138 | UDP | 7451 | 0 | 7451 | 100 | 0 | 1.5E+08 | 205.0078 | 0 | 0 | 4.559653 |
| 2 | 10.0.2.15 | 10.0.2.255 | 137 | 137 | UDP | 11418 | 0 | 11418 | 100 | 0 | 1.43E+08 | 50 | 0 | 0 | 0 |
| 3 | 10.0.2.15 | 186.50.13 | 15931 | 21963 | UDP | 9 | 0 | 9 | 100 | 0 | 6123243 | 72 | 0 | 0 | 0 |
| 4 | 10.0.2.15 | 186.18.3.7 | 15931 | 12054 | UDP | 19 | 0 | 19 | 100 | 0 | 6720635 | 108.8947 | 0 | 0 | 55.91525 |
| 5 | 10.0.2.15 | 82.231.13 | 15931 | 20205 | UDP | 17 | 0 | 17 | 100 | 0 | 6649121 | 99.17647 | 0 | 0 | 49.30963 |
| 6 | 10.0.2.15 | 78.146.22 | 15931 | 16912 | UDP | 9 | 0 | 9 | 100 | 0 | 6130551 | 72 | 0 | 0 | 0 |
| 7 | 10.0.2.15 | 82.216.40. | 15931 | 28172 | UDP | 9 | 0 | 9 | 100 | 0 | 6380580 | 72 | 0 | 0 | 0 |
| 8 | 10.0.2.15 | 95.233.19 | 15931 | 12498 | UDP | 9 | 0 | 9 | 100 | 0 | 6380364 | 72 | 0 | 0 | 0 |
| 9 | 10.0.2.15 | 151.33.12 | 15931 | 22820 | UDP | 9 | 0 | 9 | 100 | 0 | 6380308 | 72 | 0 | 0 | 0 |
| 10 | 10.0.2.15 | 79.8.8.206 | 15931 | 11030 | UDP | 9 | 0 | 9 | 100 | 0 | 6383902 | 72 | 0 | 0 | 0 |
| 11 | 10.0.2.15 | 125.237.8. | 15931 | 29135 | UDP | 9 | 0 | 9 | 100 | 0 | 6383113 | 72 | 0 | 0 | 0 |
| 12 | 10.0.2.15 | 77.165.95. | 15931 | 20852 | UDP | 9 | 0 | 9 | 100 | 0 | 6382987 | 72 | 0 | 0 | 0 |
| 13 | 10.0.2.15 | 2.192.219. | 15931 | 21159 | UDP | 9 | 0 | 9 | 100 | 0 | 6382869 | 72 | 0 | 0 | 0 |
| 14 | 10.0.2.15 | 108.28.16 | 15931 | 21272 | UDP | 9 | 0 | 9 | 100 | 0 | 6383411 | 72 | 0 | 0 | 0 |

**Sample Flow info CSV**

## ➔ Feature Extraction

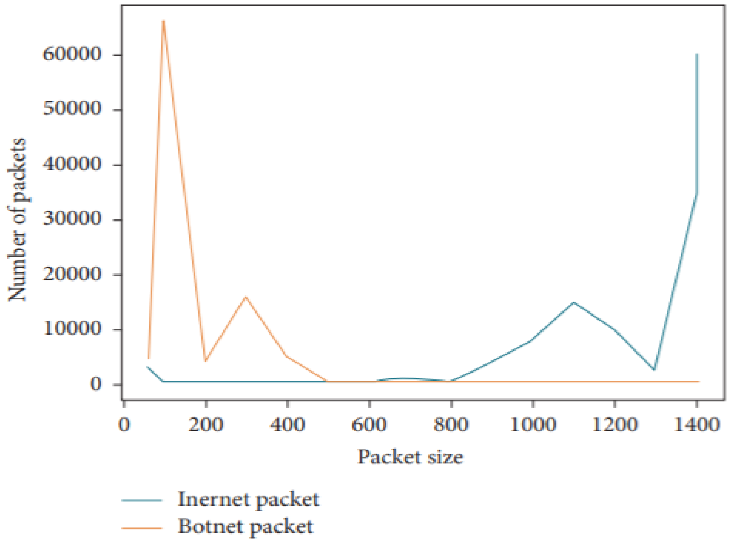
Any machine learning technique requires features for classification. We have extracted a set of 22 features from each flow. The extracted features are based on payload, duration, flow characteristics and time-based rates. The details of the features extracted is given below.

| **FEATURE** | **DATA** |
|---|---|
| Source Port | Integer |
| Destination Port | Integer |
| Protocol | String |
| Duration | Float |
| First Packet Length | Integer |
| Total Number of Bytes | Integer |
| Average Payload Length | Integer |
| Ratio of packets with same length (DPL) | Float |
| Standard Deviation of Payload Length | Float |
| Number of small Packets | Integer |

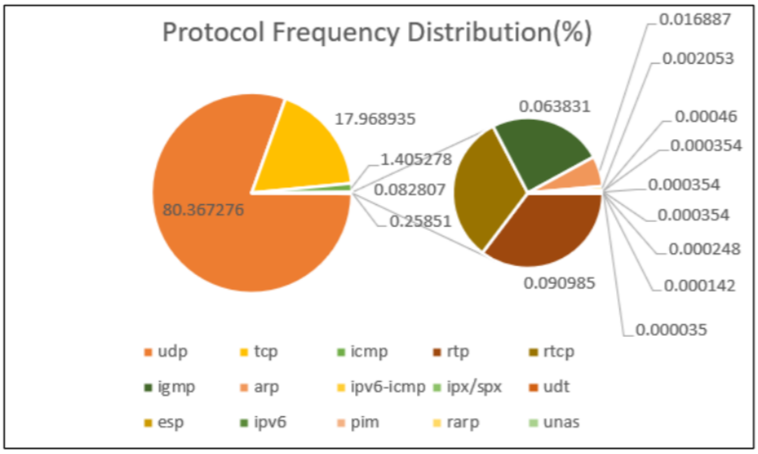
| | |
|---|---|
| Number of Null Packets | Integer |
| Percentage of small packets | Float |
| Percentage of null packets | Float |
| Average Inter-arrival Time | Float |
| Packets per second | Float |
| Bytes per second | Float |
| Number of packets exchanged | Integer |
| Number of packets received | Integer |
| Number of packets sent | Integer |
| Average packet length received | Integer |
| Average Packet Length sent | Integer |

Here small packets are packets that are smaller than a threshold (5 Mb). Packet size is a very important feature for botnet detection. The analysis of various features has been presented.

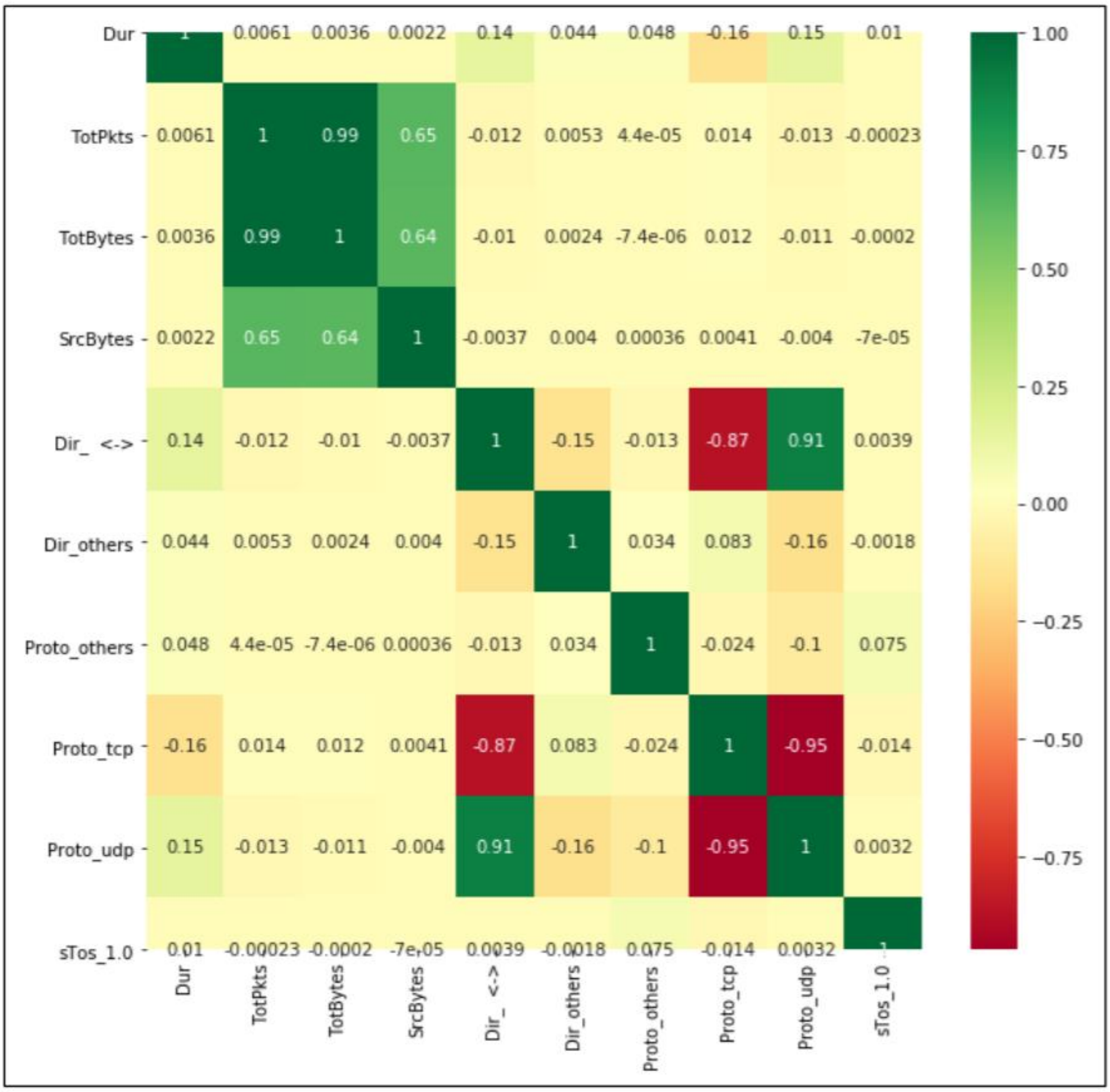
**Packet Size on Malicious and Benign Flows**



**Distribution of Protocol in Data**



**Correlation Matrix of the feature set**

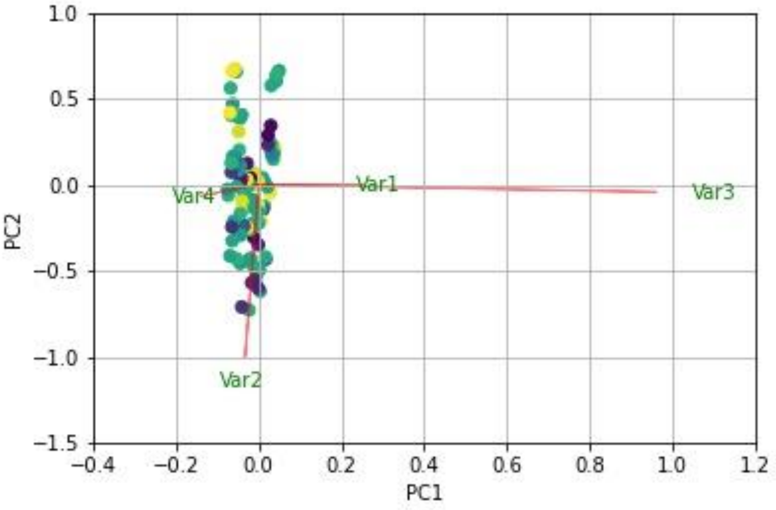


## ➔ Feature Selection

Features describe the entities to be classified but a larger number of features need not mean a better performing model. Features usually contain many redundancies and irrelevant features that negatively affect the performance of a machine learning model. To overcome this, we have used various feature selection techniques. These are stated below –

- Co-relation matrix of the features were analysed, features with correlation values close to 1 or -1 are highly correlated showing that taking both the features is a feature set redundancy.
- PCA loading scores. PCA creates principal components to be used instead of features. These are linear combinations of the features and reduce dimensionality. The features were scored based on their loading values on the first 2 Principal components. Higher loading values means that the feature has a high contribution to the principal components.
- Some experimental analysis has also been performed.

The PCA biplot of the feature set is shown below. The feature selector chose 16 features from the initial set of 22 features.



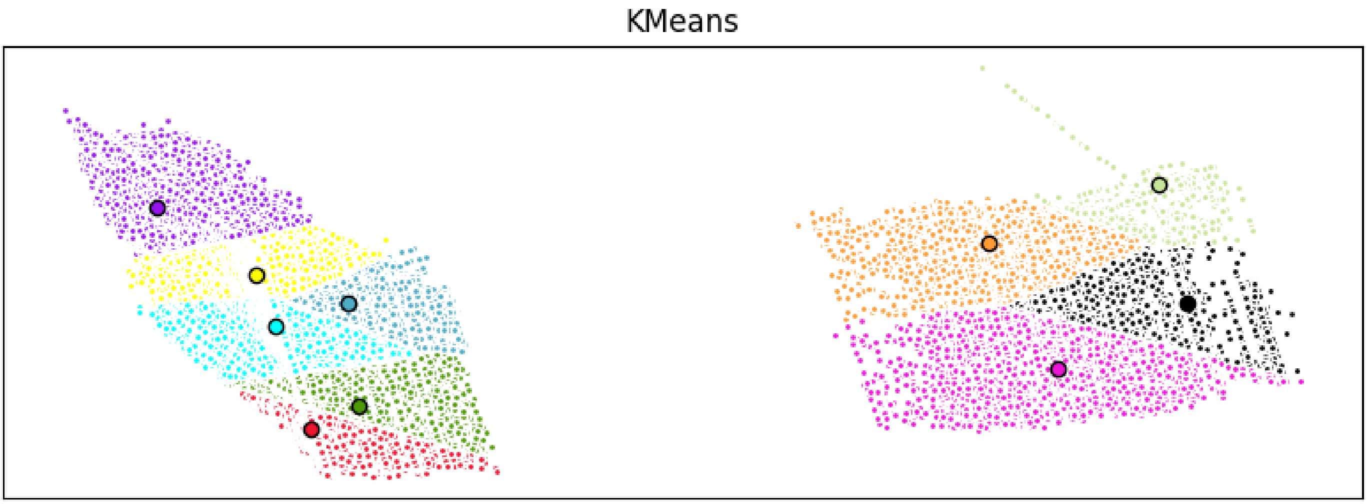
**Bi-Plot of PCA on the First 4 features of the dataset using the first 2 principal components**

## ➔ Clustering of Flows

Finally, these flow attributes can be handled by clustering algorithms, which generate different partitions on the input data. The result we need is to construct pure clusters where the input may include a little noise. That is to say, our goal is for the flows in each cluster to be from the same attack phase.

Clustering has been applied to remove dense benign and malicious clusters. Since, these can be easily removed clustering is better than direct classification. Clustering also reduces overall model training and prediction times. Results from clustering can further be used to analyse the features of the attack phase.

We have used K-Means to do an unsupervised clustering of the network traffic flows. We were able to generate 10 clusters out of which 1 was completely malicious flows and 4 were completely benign flows. Clustering is used for low level flow removal for clearly benign/malicious flows. The remaining flows have been fed to a classification model for prediction. This helps in removing the obvious benign flows first.



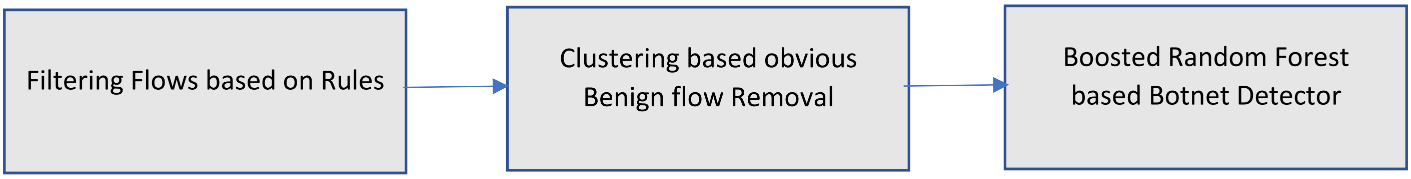
**Sample of Clustering**

### ➔ Flow Classification

The flows given by the cluster analysis are used to r=train the final classifier. We have used a Gradient Boosting Classifier with Random Forest as the base estimator. Both the GBC and Random Forest have 500 nodes each. The hyperparameters have been tuned experimentally and the results with different hyper-parameters are also provided.

Gradient Boosting ensures that the model is robust and does not over fit the data. Studies in reference have shown that boosting is the ideal approach for traffic classification

We have integrated both clustering and classification in our approach, where clustering removes the clearly benign flows and then the model classifies the remaining flows. An integrated model performs much better than any single technique.


Filtering Flows based on Rules
Clustering based obvious Benign flow Removal
Boosted Random Forest based Botnet Detector

**Network Flow Classification Summary**

### ➔ Traffic/Host Classification

For the detection of Botnet Hosts. We accumulate the traffic flow originating from a particular IP. These flows are classified using the model. A score is generated for each host. The score is simply the percentage of the host's connections that were flagged by the flow classifier.

A Threshold of 1% has been used to detect the infected hosts. Basically, if more than 1% of any host's communications are flagged as malicious, the system flags the host as a Bot. We have used traffic flows of a host to determine if the host is a bot or not.

## Experimentation and Analysis –

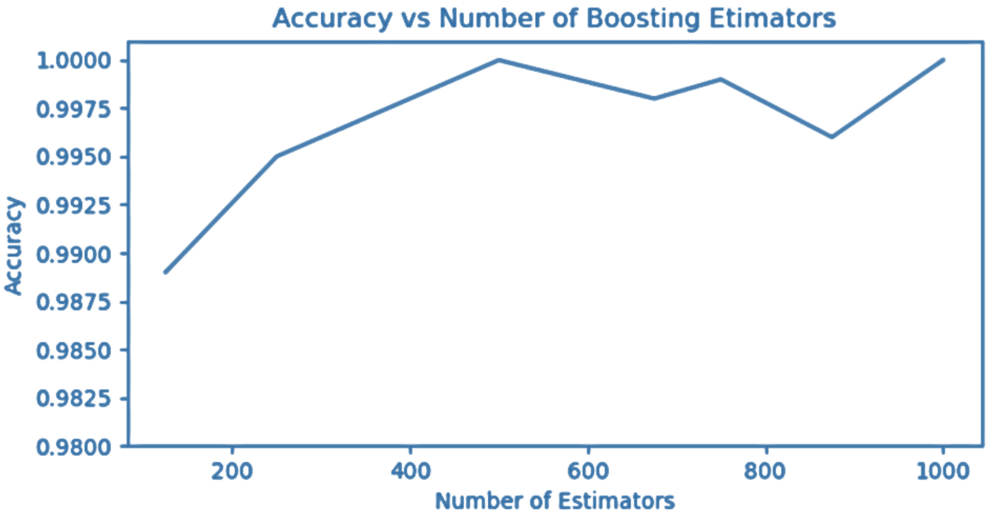
Various Analysis were conducted before finalizing the model.

### 1. Comparison of Accuracy with and without boosting

We compared the results with and without boosting. Recall has been used for evaluation instead of accuracy because the data is skewed and accuracy will not give a correct estimation of performance.

| Model | Malicious Flow Classification Recall |
|---|---|
| Random Forest | 98.17% |
| Random Forest with Gradient Boosting | 100% |

### 2. Effect of Number of Boosting Estimators on Accuracy



## 3. Comparison of Accuracy with and without Clustering

We compared the results with and without Clustering. The 2 models compared are briefly summarized below.

1. Clustering is used to remove benign clusters first. These clusters are benign traffic and contain no malicious flows. Classifier has been applied on the remaining flows.
2. Classifier has been applied directly on all the extracted flows.

Recall has been used for evaluation instead of accuracy because the data is skewed and accuracy will not give a correct estimation of performance.

| Model | Malicious Flow Classification Recall |
|---|---|
| No Clustering | 94.36% |
| KNN based Clustering | 100% |

## 4. Comparison of different Classifiers

| Model | Recall |
|---|---|
| Decision Tree | 92.16% |
| Decision Tree with Boosting | 98.63% |
| Random Forest | 98.17% |
| Random Forest with Boosting | 100% |

## 5. Effect of Feature Selection

We have used a PCA and Co-relation based feature ranking system and the best 16 features are chosen. The analysis of feature

| Feature Selector | No. of Features | Recall |
|---|---|---|
| PCA and Co-relation | 22 | 97.34% |
| PCA and Co-relation | 18 | 98.86% |
| PCA and Co-relation | 16 | 100.00% |
| PCA and Co-relation | 14 | 98.38% |

## Result and Analysis –

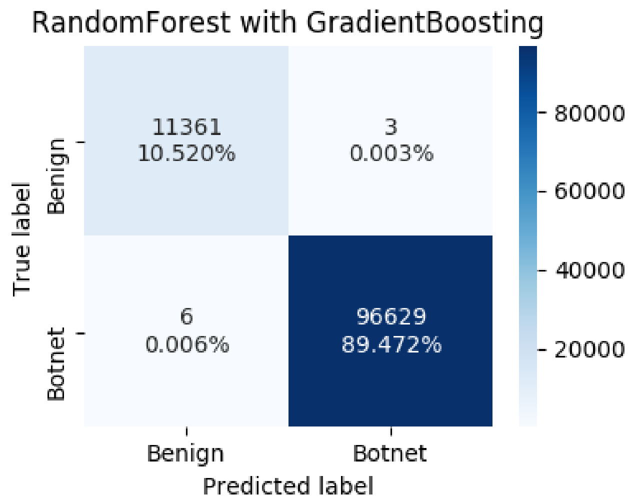
We have used a 2:1 training set – test set ratio for evaluating the performance of the model. The clustering and the classification models were trained on 67% data and the model was tested on the remaining 33%. The model was evaluated using different evaluation metrics, namely –

- Accuracy
- Precision
- Recall
- F1 Score
- True Positive Rate
- True Negative Rate
- False Positive Rate
- False Negative Rate

The results have been summarized in the following Table.

| **METRIC** | **SCORE** |
|---|---|
| Accuracy | 99.991% |
| Precision | 99.947% |
| Recall | 99.997% |
| F1 Score | 99.972% |
| True Positive Rate | 99.997% |
| True Negative Rate | 99.956% |
| False Positive Rate | 0.003% |
| False Negative Rate | 0.044% |

The confusion matrix of the model have been shown below.



| | | Actual Values | |
|---|---|---|---|
| | | Positive (1) | Negative (0) |
| Predicted Values | Positive (1) | TP | FP |
| | Negative (0) | FN | TN |

## SYSTEM AND SOFTWARE

The entire code is in Python 3 language.

The libraries used are –

1. **Wireshark (Manual PCAP Analysis)**
2. **Scapy (PCAP Parsing)**
3. joblib
4. pickle
5. numpy
6. sklearn
7. csv
8. glob
9. os
10. natsort
11. pandas
12. matplotlib
13. seaborn

The runtime of the code on an Intel i7 @ 1.8 GHz with 4 GB graphics and 15 GB RAM for a PCAP file of 100 Mb is 5 minutes.

## FINAL WORDS

We have presented a supervised plus unsupervised model based on a KNN clustering and with a Boosted Random Forest Classification for Botnet Detection using a flow-based approach. The model is highly robust and achieves a 99.999% detection rate on the given data. Further various feature selection and engineering techniques have been used to extract complex features from the data with their in-depth analysis.

## REFERENCES

- http://downloads.hindawi.com/journals/jcnc/2020/9024726.pdf
- https://ieeexplore.ieee.org/abstract/document/7997375
- https://www.researchgate.net/publication/259117704_Botnet_detection_based_on_traffic_behavior_analysis_and_flow_intervals
- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7361120
- https://vnetman.github.io/pcap/python/pyshark/scapy/libpcap/2018/10/25/analyzing-packet-captures-with-python-part-1.html
- https://reader.elsevier.com/reader/sd/pii/S187705091830869X?token=06A4B3BDB78316127155F4BC913398993104C8C320CFC9DDC4D4C72B7DB983E013DED3F57872E1A7D51DC9D5DE9BAF9F
- https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1917&context=etd_projects
- https://link.springer.com/chapter/10.1007/978-3-642-30436-1_8