Darknet Traffic Classification using Machine Learning

Aslam Baig, Siva Sai, Sumana Vemuri, Karthik Ramanarayana, Suresh Devi

University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250

(Dated: December 16, 2021)

Abstract

Darknet traffic classification is significantly important task to categorize real-time applications in an encrypted network. Monitoring the darknet traffic to identify content, services, and applications is a high demand research topic in network traffic control systems. In the case of modern firewalls, decryption of packets happens which doesn't appeal for private users which might be treat for privacy breach. Hence, identifying any information from an encrypted traffic is a highly challenging task. There are many traditional techniques to classify internet traffic like Port Based, Pay Load Based and Machine Learning Based technique. Out of which Machine Learning techniques successfully classify the darknet traffic data due to its high reliability. In this paper, we propose a robust classification technique built on two encrypted traffic datasets merged to create a darknet dataset. This involves high level feature extraction from network packet data then training a robust machine learning classifier for traffic identification based on features such as packet payload, inter-arrival time sequence etc. Three Supervised machine learning Classifiers Decision tree, Random Forest and KNN classifiers are applied to characterizes darknet traffic. Experimental analysis shows that Random Forest classifier gives good accuracy result as compare to other classifies.

I. INTRODUCTION

Darknet is the unused address space of the internet. Any communication from the dark space is considered skeptical owing to its passive listening nature. Due to the absence of legitimate hosts in the darknet, any traffic is contemplated to be unsought and is characteristically treated as probe, backscatter, or misconfiguration. So, Darknet traffic classification is significantly important to categorize real-time applications from illicit malicious activity software. Analyzing darknet traffic helps to combat alleged activities before that assault the cyber world, in early monitoring of malware before onslaught and detection of malicious activities after outbreak. This classification also helps to improve the Quality of Service (QoS) for internet users by allocating the bandwidth and resources without breach of privacy violations.

In recent years, numerous network traffic classification techniques have been proposed to classify unknown classes. The first one is Port Based Technique. It is a great technique for network traffic classification/ identification. This technique includes a port, which is firstly registered in Internet Assign Number Authority (lANA). But this technique failed due to increase of Peer-to-Peer applications (P2P), which use dynamic port numbers. Dynamic port number means unregistered number with Internet Assign Number Authority.

Then second one is Payload Based technique also called Deep Packet Inspection (DPI) technique, but it cannot be used for encrypted data network applications as numerous network applications use encrypted techniques to protect data from detection. So, this approach also failed due to use of encrypted flow of applications. Hence, the above two techniques don't help to solve the purpose of the classifying the encrypted traffic.

Thus, non-biased and statistical based techniques are required to handle this problem. The Supervised Machine Learning algorithms have the capability to classify the classes based on the different features passed as input to it. Hence, traffic features in a network serves as input to classify the traffic into Benign or malicious as well as to know what type of applications flow in the encrypted network.

Therefore, in this project, ML-based classification models were used to classify the traffic. The proposed architecture uses existing network statistics for understanding darknet traffic patterns to the classify incoming network traffic in a darknet into the respective classes (type of applications).

The rest of the paper is structured as, Section II introduces the Related work and Literature review. Section III demonstrates the proposed Darknet Classification Model. Section IV presents the

experimental result of the system. Section V represents the Future scope and Section VI concludes this paper.

II. Literature Review & Related Work

A. Port based Network Classification:

Traditionally, in Port-based network classification technique, a classification of network applications is executed using the well-known ports number. Moreover, the first network applications are registered in their ports on the Internet Assigned Number Authority (AINA). In this way, the traffic is identified corresponding to the registered ports number registered in AINA. Table 1.1 shows different types of applications and its ports numbers assigned by AINA. For instance, E-mail applications use 25 (SMTP) port number to send emails and to receive email 110 (POP3) port is used. In this way, web applications use 80 ports number.

TABLE 2.1 ANA ASSIGNED PORT NUMBER

Assigned Port	Application
20	FTP Data
21	FIP
22	SSH
23	Telnet
25	SMTP
53	DNS
80	HTTP
110	POP3
123	NTP
161	SNMP
3724	WoW

Thus, this technique does not provide good classification accuracy results and fails due to using dynamic port number of new applications to evade being detected.

B. Payload based Network Classification:

In this method, the contents of the packets are examined looking characteristics signatures of the network applications in the traffic. This is the first alternative to ports-based method by Karagiannis et al. This technique is specially proposed for Peer to Peer (P2P) applications. It means applications which use dynamic port number to identity traffic in a network. Below is the table, which illustrates examples, used by Karagiannis et al.

TABLE 2.2. KARAGTANNIS DESCRIBE STRINGS AT THE BEGINNING P2P PROTOCOL PAYLOAD

P2P Protocol	String	Trans. Protocol
Edonkey 2000	0xe319010000	TCP/UDP
	0xe53f010000	
Fasttrack	"Get /.hash"	TCP
	0x2700000002980	UDP
BitTorrent	"0x13Bit"	TCP
Gnutella	"GNUT" "GIV"	TCP
Aress	"GET hash"	UDP
	"Get Shal"	

There are some drawbacks with this technique, it is very expensive hardware for pattern searching in a payload. The second problem in this technique is that it does not work in encrypted network application traffic. Finally, this approach needs continuous update of signature pattern of new applications.

C. Machine Learning Based Classification.

Machine learning (ML) based classification technique is based on data set (Labelled Data Set) where a machine learning classifier is trained as input and then using the trained sample prediction, unknown classes are classified using Supervised Machine Learning Models. This supervised classification infers function from labelled training data set. This method starts with a training dataset TS.

$$TS = \langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \ldots\ldots\ldots \langle x_N, y_M \rangle,$$

Such that Xi is the feature vector which belongs to i^{th} and y_i is its output predicted value.

Some related traffic classification models were proposed using ML based classifiers.

In 2013, R. *C. Jaiswal and S. D. Lokhande,* have used the ML algorithm for classifying network traffic by application. They have trained few ML models using labelled data by applications such as Post Office Protocol 3 (POP3), Skype, Torrent, Domain Name System (DNS), Telnet were recognized by the classifier. For this experiment, they have tested six different classification models and compared accuracy.

A. Y. Nikravesh, S. A. Ajila, C. Lung and W. Ding have experimented with mobile network traffic classification ML models. In their project, there are three main objectives. Comparing the accuracy of three classification models [SVM, Multi-Layer Perceptron with Weight Decay (MLPWD), MLP]. Analysing the effect on accuracy by varying the size of the sliding window. Comparing the accuracy of predictions of the models for unidimensional /multi-dimensional datasets. In their project, they have selected 24 features and selected one of the features as the target to predict. In terms of accuracy, the paper has concluded that in multi-dimensional data sets SVM performs better and in

unidimensional data sets, the MLPWD model performs better.

P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares and H. S. Mamede, they have experimented with the data collection and traffic classification process in software defined networks. In their work, they have developed a network application to collect OpenFlow data in a controlled environment. Only Transmission Control Protocol (TCP) traffic was considered for this project. Several packets of information were gathered using different methods. For example, Packet IN messages were used to extract source/destination IP addresses and port addresses. First five packet sizes and timestamps were collected from the controller since in this experiment the next five packets after the initial handshake between server and client ow through the controller. Flow duration was collected by subtracting the timestamp of the initial packet and the time stamp of the message received by the controller regarding the removal of the row entry. To avoid the high variance of the data set, they have used a scaling process named standard score. They have also mentioned that highly correlated features are not contributing much to the algorithm but increase the complexity in computation. They have used the Principle Component Analysis (PCA) algorithm to remove these high correlated factors. Random Forest, Stochastic Gradient Boost, Extreme Gradient Boost are the classifiers used in their research. The results were compared by evaluating the accuracy of each label.

Previously, training supervised Naive Bayes classifiers as header-driven discriminators achieved high accuracy for many network traffic systems. However, due to the growth of encrypted traffic, such approaches have been rendered ineffective. More recent approaches have focused on application-level identification without using IP addresses, port numbers, or decrypted payload information.

Alshammari et al. find that Decision Trees achieve the best accuracy when classifying Skype and SSH traffic. *Okada et al*. improve on this work by focusing on the creation of statistical features related to packet size and packet transfer times for application classification (FTP, DNS, HTTP, etc.). These features achieve high accuracy when paired with Support Vector Machines classifiers.

In the paper "Towards Early Detection of Novel Attack Patterns through the Lens of A Large-Scale Darknet "by Tao Ban, Shaoning Pang, Masashi Eto, Daisuke Inoue, Koji Nakao, and Runhe Huang. Researchers provided pattern of the attacks in a darknet, and the applications affected due to unable to detect the type of malicious software masquerading as normal applications

Sikha Bagui, Xingang Fang, Ezhil Kalaimannan, Subhash C. Bagui, and *Joseph Sheehan*. 2017 provided study of classification of VPN network traffic flow using time-related features. In this paper they proposed the performance of ML models on the VPN based networks.

All these Network traffic classifiers were built on top of the traffic collected from Open network. There are only significant efforts on detecting encrypted traffic covering either VPN or Tor traffic separately. None of the papers attempted to combine VPN and Tor traffic in a single dataset that covers a wide range of captured applications and hidden services provided by darknet.

Though the real time classification rate of aforementioned models is promising when the sample from the Open network across various platforms, the models must also need to classify the traffic from the darknet in the same way with much accuracy.

Thus, Classification of the Darknet traffic is a quite challenging research problem where we are trying to predict the type of application which was used under both open network and darknet.

III. PROPOSED MODEL

The existing solutions and related research is based on the classification of the encrypted traffic flow in the open network or VPN based network, whereas we are trying to classify the type of the application in a darknet in the both cases of the VPN and non-VPN, tor and Non-Tor access scenarios.

With the motivation of the high classification rate of the Supervised learning algorithms on the traffic from the Open Network. In this Proposed solution, those models are built and evaluated on top of the amalgamation of Darknet data and Open Network data. This darknet Classifier is then rendered over the testing samples to evaluate the performance.

Building a classifier model is a stepwise approach where each step represents the phases of the Machine Learning project pipeline.

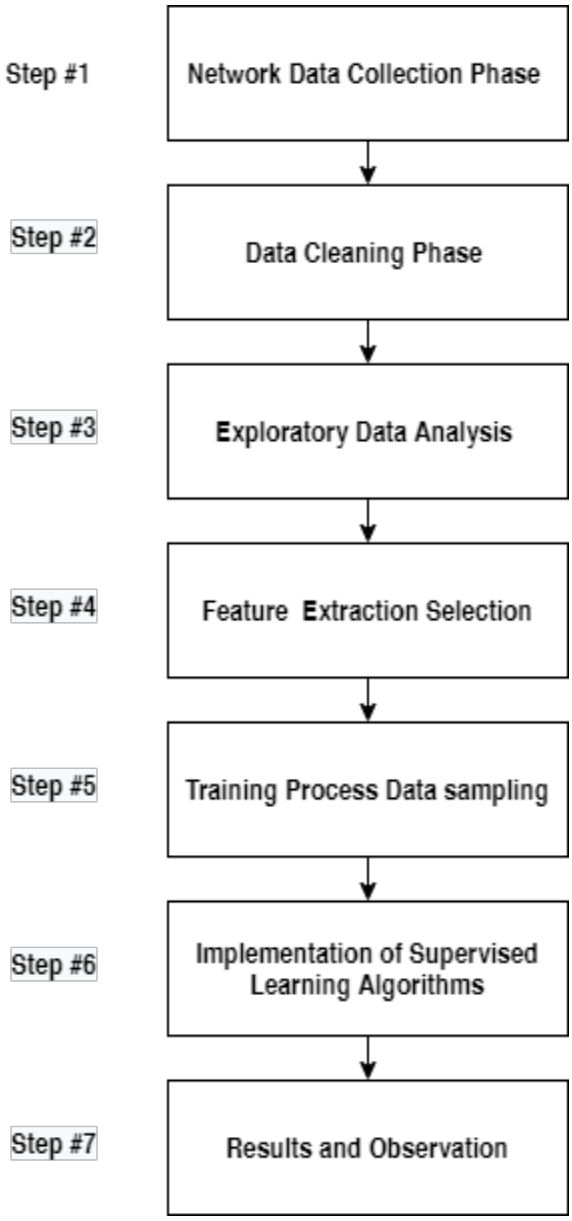

FIG 3.0 PHASES IN BUILDING DARKNET CLASSIFER

PHASE I. DATA COLLECTION

In this project, the darknet dataset is gathered from the UNB's research-based hybrid datasetCICDarknet2020. It is amalgamated two publicly available datasets (ISCXVPN2016 & ISCXTor2016) to create a complete darknet dataset, named CIC-Darknet2020, covering VPN and Tor traffic. The CICDarknet2020 data set contains 84 features and 141531 samples Each row represents a traffic flow from a source to a destination and each column represents features of the traffic data.

I. DATASET CICDARKNET2020

In CICDarknet2020 dataset, a two-layered approach is used to generate benign and darknet traffic at the first layer. The darknet traffic constitutes **Audio-Stream, Browsing, Chat, Email, P2P, Transfer, Video-Stream and VOIP** which is generated at the second layer. To generate the representative dataset, UNB amalgamated datasets ISCXTor2016 & ISCXVPN2016, and combined the respective VPN and Tor traffic in corresponding Darknet categories.

The below figures 3.1.1 and 3.1.2 represents the details of number of samples of benign and darknet traffic at first layer and highlights the number of encrypted flows in our darknet traffic.

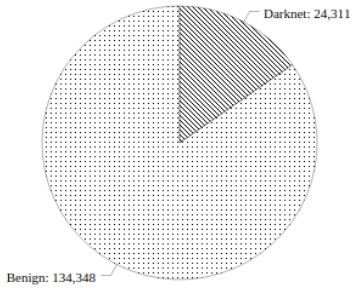

FIG 3.1.1 LAYER 1

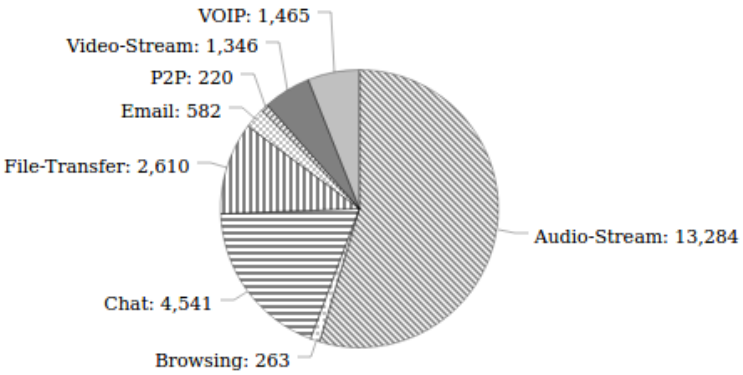

FIG 3.1.2 LAYER II

II. FEATURES & CLASSES IN THE DATASET

This dataset constitutes 84 features that better defines the categories of applications used the darknet which forms our classification labels. The below table represents the 6 output labels in our dataset.

Traffic Category	Applications used
Audio-Stream	Vimeo and Youtube
Browsing	Firefox and Chrome
Chat	ICQ, AIM, Skype, Facebook and Hangouts
Email	SMTPS, POP3S and IMAPS
P2P	uTorrent and Transmission (BitTorrent)
Transfer	Skype, FTP over SSH (SFTP) and FTP over SSL (FTPS) using Filezilla and an external service
Video-Stream	Vimeo and Youtube
VOIP	Facebook, Skype and Hangouts voice calls

FIG 3.1.3 OUTPUT CLASSES IN DATASET

This dataset was a perfect match for our objectives and satisfy all the three main components of a good dataset, which are real-world, substantial and diverse.

PHASE II. DATA CLEANING

This raw dataset has 84 features and 141531 samples, out of which some inconsistencies such as missing data and erroneous samples are present. These kinds of impurities were removed by cleansing the data by dropping them to avoid biasing and inconsistences in model building. For example, missing and duplicate data causes biasing and inconsistencies in the models. Hence, these duplicates and missing values need to be identified and remove from the dataset or filled with value values close to the mean of that feature.

In this dataset, there are several features contains different data types. But some ML models can only work with numeric values. To use those data types for the ML model training, it is necessary to convert or reassign numeric values to represent its correlations with other features.

PHASE III. EXPLORATORY DATA ANALYSIS

The pruned dataset is then analysed across identifying underlying relationships among each feature in the dataset. The highly correlated values and outliers are detected and removed from the dataset. First, we calculate the correlation matrix based on Pearson's correlation coefficients for a sample of n observations. Given a series of n measurements between features *xi* and *xj*. the sample correlation coefficient *r(xi,xj)* between *xi* and *xj* is given by

$$r_{x_i x_j} = \frac{\sum_{\alpha=1}^{n}(x_{i,\alpha} - \bar{x}_i)(x_{j,\alpha} - \bar{x}_j)}{\sqrt{\sum_{\alpha=1}^{n}(x_{i,\alpha} - \bar{x}_i)^2 \cdot \sum_{\alpha=1}^{n}(x_{j,\alpha} - \bar{x}_j)^2}},$$

FIG 3.3.1 CORRELATION COEFFICIENT

where *xim* and *xjm* represent the arithmetic means of *xi* and *xj*, respectively.

Then, highly corelated values which have the value of *r(xi,xj)* > 9 (optimum value 0.8) are deleted from the dataset . For this data set, there are 24 highly correlated features which need to be removed from to avoid multi collinearity.

```
#deleting columns which have r-value>9
col_cor = set()
for i in range(len(cor.columns)):
    for j in range(i):
      if (abs(cor.iloc[i,j])>=0.8)) and (cor.columns[j] not in col_cor):
          col_name = cor.columns[i]
          col_cor.add(col_name)


print("cols are: ",col_cor)
print("no of features: ", len(col_cor))
```

```
cols are:  {'ACK Flag Count', 'Flow IAT Max', 'Idle Min', 'Subflow Bwd Bytes', 'Fwd Packets/s', 'Fwd Segment Size Avg', 'Bwd Packets/s', 'Bwd Segment Size Avg', 'Fwd Packet Length Std', 'Fwd IAT Mean', 'Packet Length Std', 'Average Packet Size', 'Idle Max', 'Bwd Packet/Bulk Avg', 'Packet Length Max', 'Fwd IAT Total', 'Total Length of Bwd Packet', 'Bwd Header Length', 'Bwd IAT Total', 'Fwd IAT Max', 'Fwd Header Length', 'Bwd IAT Mean', 'Subflow Fwd Bytes', 'Bwd Packet Length Std'}
no of features:  24
```

FIG 3.3.2 Highly Correlated features

Second, this dataset is set to free from Outliers using Outlier treatment. One of the notable methods is using Inter Quartile Range (IQR) to remove the Outliers in the dataset. For a given dataset the maximum distribution of features falls under the IQR i.e., between Q1 an Q3.

$$IQR = Q_3 - Q_1$$

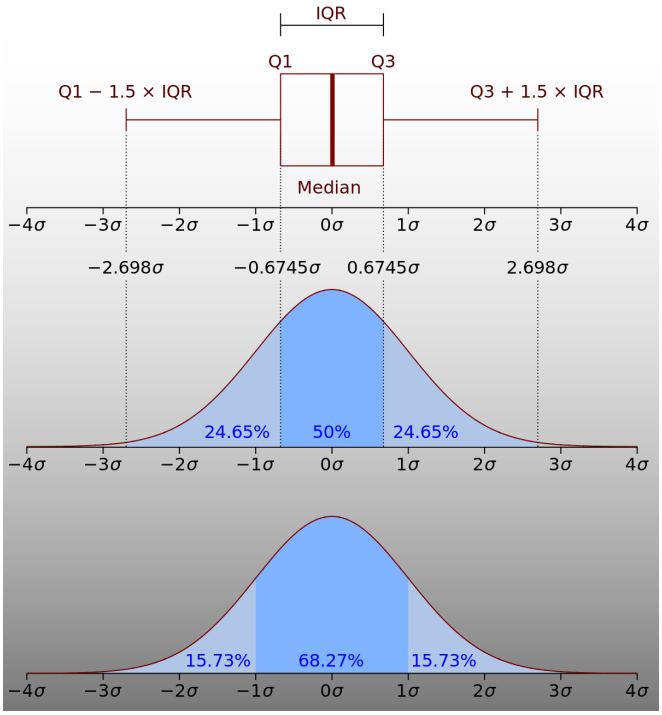

FIG 3.3.3 INTER QUARTILE RANGE

Thus, the data points beyond the IQR are considered as the outliers and can removed from the dataset by using the below formula.

```
#Finding inter quartile range
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3-Q1
print("The IQR of all data: ",IQR)
```

```
#Dropping the outliers
df_new = df[~((df < (Q1 - 1.5*IQR)) | (df > (Q3 + 1.5*IQR))).any(axis=1)]
df_new.head()
```

FIG 3.3.4 DELETION OF VALUES BEYOND IQR

PHASE IV. FEATURE EXTRACTION & SELECTION

The most important factors that concerned when selecting features were relatability to the research objective and should easily be accessible by the models without much complexity in identifying underlying patterns. The features are extracted by dropping the unnecessary and selecting the best features that have high importance for classification.

A) Dropping Least useful features:

While extracting and selecting the useful features, the features that are irrelevant for our objective must be dropped by manually selection. Selected features to drop from dataset are as follows: ['Flow ID','Src IP','Dst IP','Timestamp']

B) Eliminating the Contextual Anomalies in data:

Contextual Anomalies might cause inconstancies in the Prediction and must be handled by detecting using the tree based unsupervised Isolation Forest algorithm. Minority classes and the attribute-values which differs from standard normal distribution are eliminated.

For n instances in a dataset, anomaly score (s) of an instance (x) can be defined by using below formula

$$s_{(x,n)} = 2^{-\frac{E(h(x))}{c(n)}}$$

FIG 3.4.1. Anomaly Score

Where, c(n) is the average of h(x) given n, we use it to normalize h(x).

```
from sklearn.ensemble import IsolationForest

random_state = np.random.RandomState(42)

model=IsolationForest(n_estimators=100,max_samples='auto',contamination=float(0.2),random_state=random_state)

for col in X.columns:
        model.fit(X[[col]])
        print("Result of " + col + ":")
        print(model.get_params())
```

FIG 3.4.2. ISOLATION FOREST IMPLEMENTATION

C) Handling Imbalanced dataset using SMOTE:

Too many minority classes can cause problem of imbalanced classification. A problem with imbalanced classification is that there are too few examples of the minority class for a model to effectively learn the decision boundary. Hence, to mitigate this, Synthetic Minority Oversampling Technique is used to oversample the examples in the minority class. SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbours. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b.

Hence, the training dataset is free from the minority classes to avoid imbalanced classification.

```
# Splitting the dataset into test train split
X_train, X_test, y_train, y_test = train_test_split(Xn, Yn, test_size=0.20, shuffle=True)

#applying SMOTE (Synthetic Minority Oversampling Technique) to overcome the imbalanced classification by to oversampl
smote= SMOTE('auto')
X_sm,y_sm = smote.fit_resample(X_train, y_train)
print(np.shape(X_sm), np.shape(y_sm))

# saving the output labels in a dictionary format
dic = {}
for i in y_sm:
    if i in dic.keys():
        dic[i]+=1
    else:
        dic[i]=1
print(dic)
```

```
(9014, 56) (9014,)
(45689, 56) (45689,)
{5: 6527, 4: 6527, 0: 6527, 2: 6527, 1: 6527, 6: 6527, 3: 6527}
```

FIG 3.4.3. SMOTE

D. Selecting Useful features using SelectKBest Class

Selection of n best features makes model more interpretable and improves the model training rate. Chi-squared (chi^2) statistical techniques for non-negative features better defines the best useful features to be selected.

The Chi-Squared statistics are calculated using the following formula where "O" stands for observed or actual and "E" stands for expected value if these two categories are independent. If they are independent these O and E values will be close and if they have some association then the Chi-squared value will be high.

$$\chi^2 = \frac{(O_{11} - E_{11})^2}{E_{11}} + \frac{(O_{12} - E_{12})^2}{E_{12}} + \cdots + \frac{(O_{mn} - E_{mn})^2}{E_{mn}}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

FIG 3.4.4: Chi -Squared Formula

By implementing Chi-squared statical method with SelectKBest class, top best features can be selected from the given features list.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

X_ordered_rank_feat = SelectKBest(score_func=chi2, k='all')
X_ordered_feat= X_ordered_rank_feat.fit(X_sm, y_sm)
```

```
dfscores=pd.DataFrame(X_ordered_feat.scores_,columns=["Score"])
dfcolumns=pd.DataFrame(X_col_names)
features_rank=pd.concat([dfcolumns,dfscores],axis=1)

# Feature vs score i.e its effect on target variable
features_rank.columns=['Features','Score']
features_rank.sort_values(by=['Score'],ascending= False)
```

	Features	Score
53	Idle Mean	2.388827e+18
3	Flow Duration	6.925753e+09
18	Fwd IAT Std	5.982073e+09
16	Flow IAT Std	5.702904e+09
15	Flow IAT Mean	2.840596e+09
1	Dst Port	8.858334e+08
19	Fwd IAT Min	6.493460e+08
17	Flow IAT Min	2.984355e+08
45	FWD Init Win Bytes	2.282206e+08
0	Src Port	4.677719e+07

FIG 3.4.5: Implementation of best features from Chi -Squared Formula

E. Selecting best Features using Feature Importance.

Alternatively, inbuilt class feature_importances of tree-based classifiers such as ExtraTree Classifier is used for extracting the top n. features from the dataset used for the classification.

From the available ordered features, Top 20 features are considered for the classification than that of the others.

```
#Feature importance is an inbuilt class that comes
#we will be using Extra Tree Classifier for extrac
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X_sm,y_sm)
```

```
ExtraTreesClassifier()
```

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index= X_col_names)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```

FIG 3.4.6: Feature Importance using Extra Tree Classifier

The below graph represents the feature importance of the top 20 useful features in the pruned dataset.

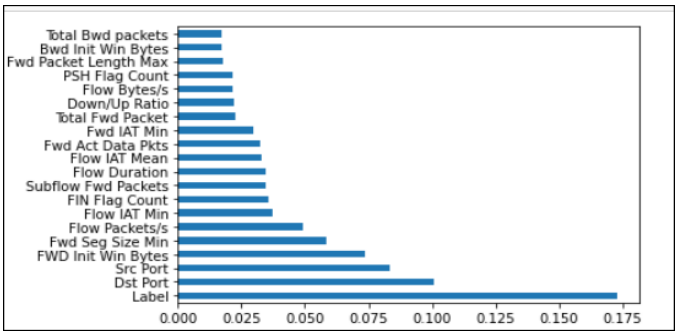

FIG 3.4.7: Feature importance of top 20 features

Once, the features are selected, The Final list of the statistical features are prepared for the model building. The below list represents the most useful features list used for our Darknet Traffic classification.

Feats *= ['Label','Dst Port','Fwd Seg Size Min','Src Port','FWD Init Win Bytes','Total Fwd Packet','Bwd Init Win Bytes','Flow Duration', 'Flow IAT Mean','Total Bwd packets', 'FIN Flag Count','Flow Bytes/s','Packet Length Mean','Total Length of Fwd Packet','Fwd Packet Length Mean','Down/Up Ratio','Fwd IAT Std','Packet Length Variance']*

PHASE V. TRAINING PROCESS SAMPLING

The filtered dataset is set to Model training by splitting into the Train and test split which a relevant trade off of 70 % and 30%. The test split is then used for validation and evaluation of the performance of the models.

PHASE VI. IMPLEMENTING THE ALGORITHMS ON PREPARED DATASET.

The Training data split was rendered over the multiple classification models available, and their performance is evaluated over the training and test splits. each and every model classify data with different mathematical models. Some models could perform better, and some models perform poorly. Out of such models, a best classifier is selected based on the evaluation metrics.

For our objective, a multi class classification problem, There are 3 mostly used renowned supervised models available. they are

- *Decision Tree*
- *Kth Nearest Neighbour*
- *Random Forest*

A. Decision Tree Classifier

It is one of the best supervised learning models that classifies data based on information gains by calculating the entropy of the dataset. It is a graphical representation of all the conditions and decisions of the dataset. The root node will be calculated using entropy with the highest information gain among the dataset. This process will continue to split branches and complete the tree. Each internal node is a test on attribute and branches represents the outcome. Leaf represents a class label. The decision tree can use numeric and categorical data for the classification problems. It also supports nonlinear relationships between features.

When the Decision Tree classifier is trained over the training split. It provided the score of the 98% accuracy

Decision Tree

```
print("Starting to train")
dt = DecisionTreeClassifier()
dt.fit(X_train , y_train)
```

```
Starting to train

DecisionTreeClassifier()
```

```
dt.tree_.node_count, dt.tree_.max_depth
```

```
(961, 26)
```

```
dt.score(X_test, y_test)
```

```
0.9861342207432058
```

FIG 3.6.1: Decision Tree Classifier

Classification report of the Decision Tree Classifier shows the precision recall f1 score and support metrics for each of the output labels classified by the Decision Tree. Below fig represents the Classification report.

```
y_pred = dt.predict(X_test)
print('accuracy %s' % accuracy_score(y_test, y_pred))
print("Classes: ", le.inverse_transform([0,1,2,3,4,5,6]))
print(classification_report(y_test, y_pred, target_names= le.inverse_transform([0,1
```

```
accuracy 0.9861342207432058
Classes:  ['AUDIO-VIDEO-STREAMING' 'BROWSING' 'CHAT' 'EMAIL' 'FILE-TRANSFER' 'P2P'
 'VOIP']
                     precision    recall  f1-score   support

AUDIO-VIDEO-STREAMING       0.93      0.92      0.93       106
            BROWSING       0.60      0.55      0.57        11
                CHAT       0.82      0.90      0.86        10
               EMAIL       1.00      0.85      0.92        13
       FILE-TRANSFER       0.66      0.73      0.69        26
                 P2P       1.00      1.00      1.00      1634
                VOIP       0.50      0.33      0.40         3

           micro avg       0.99      0.99      0.99      1803
           macro avg       0.69      0.66      0.67      1803
        weighted avg       0.99      0.99      0.99      1803
```

FIG 3.6.2: Classification Report of Decision Tree

The Confusion matrix of this classifier against the given output classes are better depicted from the below chart

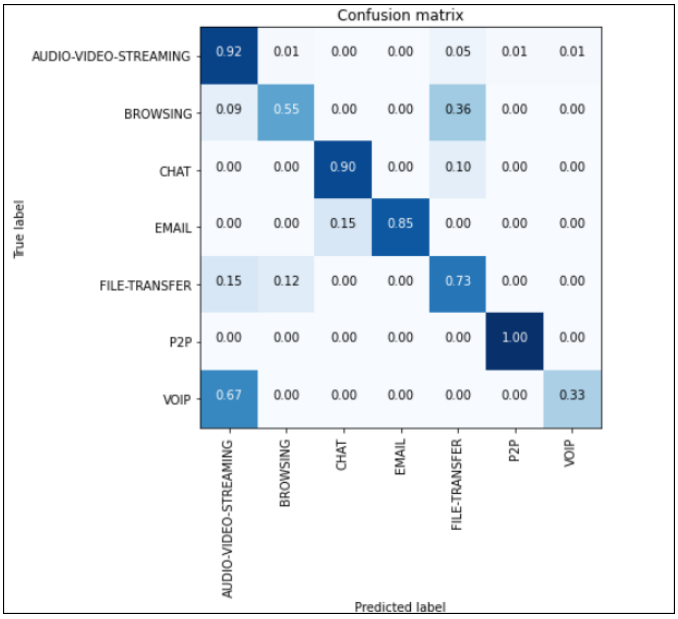

FIG 3.6.3: Confusion Matrix of Decision Tree

The Feature importance depicts the role played by each feature in predicting the output classes. The graph 3.6.4 shows the feature importance for decision Tree classifier

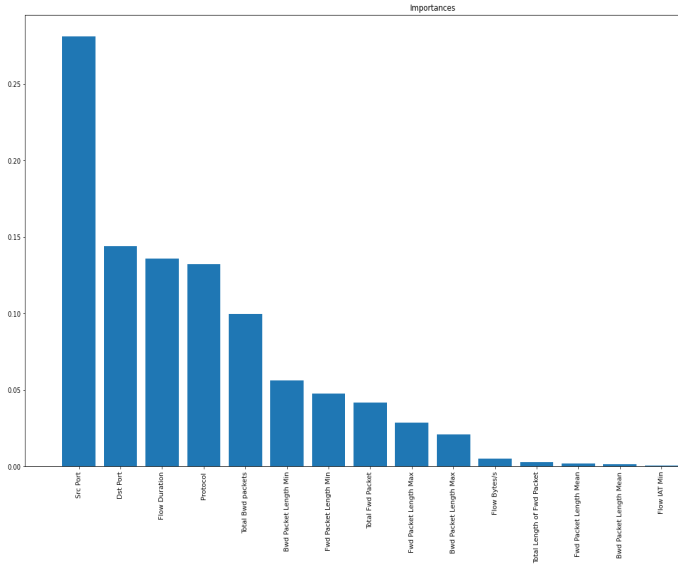

FIG 3.6.4: Feature importance of Decision Tree

B. Kth Nearest Neighbor Classifier

KNN is an instance based supervised learning algorithm which can be used for our objective. In the KNN model, the value k represents the number of neighbors needs to consider for the classification. The model will check the labels of those neighbors and select the label of the majority. The value k should be an odd number to avoid drawing the decision. It is a robust model that can work with noisy data and perform better if the training data set is large.

On comparing the accuracies of the KNN model under each K, the better suited value for K is 15. Thus, this model better performs with 15 neighbors.

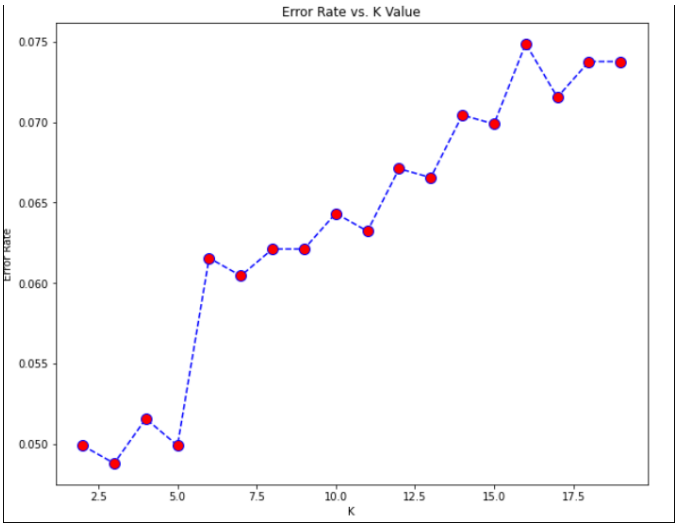

FIG 3.6.5: Error Rate vs K value

When the KNN classifier is trained over the training split. It provided the score of the 96% accuracy

```
4]: #Here we can see that that after arouns K>15 the error rate
    #just tends to hover around 0.06-0.05 Let's retrain the model with that and check the cl
    neigh = KNeighborsClassifier(n_neighbors = 15, metric='manhattan', weights='distance')

5]: neigh.fit(X_train, y_train)

5]: KNeighborsClassifier(metric='manhattan', n_neighbors=16, weights='distance')

6]: neigh.score(X_test, y_test)

6]: 0.9656128674431503
```

FIG 3.6.6: KNN Classifier

Classification report of the KNN Classifier shows the precision recall f1 score and support metrics for each of the output labels classified by the KNN. Below fig 3.6.7 represents the Classification report.

```
: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.88	0.77	0.82	108
1	0.40	0.43	0.41	14
2	0.79	0.73	0.76	15
3	1.00	0.85	0.92	13
4	0.24	0.38	0.29	21
5	1.00	0.99	1.00	1631
6	0.08	1.00	0.14	1
accuracy			0.97	1803
macro avg	0.63	0.74	0.62	1803
weighted avg	0.98	0.97	0.97	1803

FIG 3.6.7: Classification Report of Knn Classifier

The Confusion matrix of this classifier against the given output classes are better depicted from the below chart

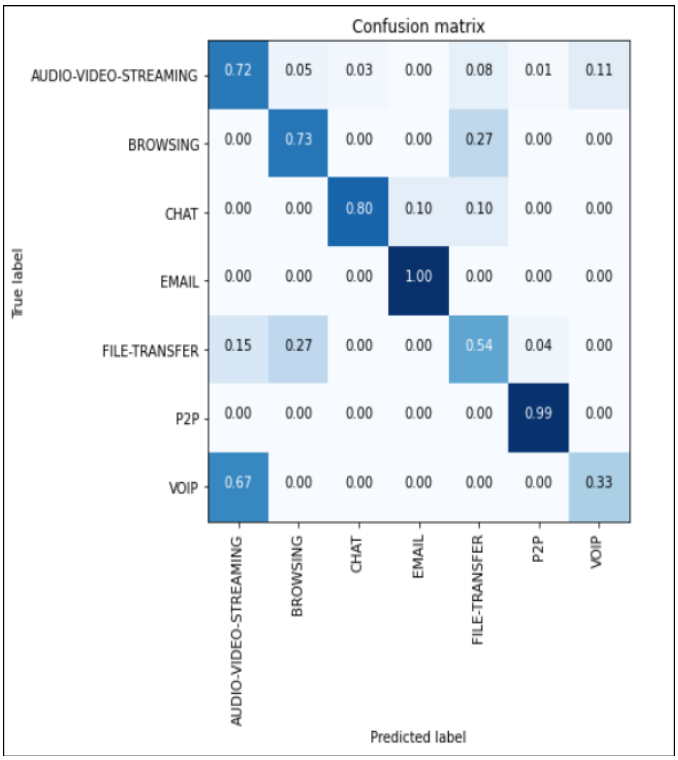

FIG 3.6.8: Confusion Matrix of Knn Classifier

The Feature importance depicts the role played by each feature in predicting the output classes. The graph 3.6.8 shows the feature importance for Knn classifier

```
n_feats = X_train.shape[1]

print('Feature  Importance')
for i in range(n_feats):
    X = X_train[:, i].reshape(-1, 1)
    scores = cross_val_score(neigh, X, y_train, cv=3)
    print(f'{i}        {scores.mean():g}')
```

```
Feature  Importance
0        0.426558
1        0.484996
2        0.302437
3        0.553262
4        0.587647
5        0.309812
6        0.3737
7        0.574493
8        0.592112
9        0.310381
10        0.169932
11        0.329861
12        0.337565
13        0.325417
14        0.288582
15        0.304537
16        0.282738
17        0.280089
```

FIG 3.6.8: Feature importance of Knn Classifier

C. Random Forest Classifier

It is one of the powerful supervised learning algorithms, which can perform both regression and classification problems. This is a combination of multiple decision tree algorithms and higher the number of trees, higher the accuracy. It works as same as the decision tree, which based on information gain. In classification, each decision tree will classify the same problem and the overall decision will be calculated by considering the majority vote of the results. The most important advantage of this model is that it can handle missing values and able to handle large datasets.

When the Random Forest classifier is trained over the training split. It provided the score of the 99% accuracy

Random Forest

```
: #rf = RandomForestClassifier(max_depth=60
rf = RandomForestClassifier()
rf.fit(X_train , y_train)
```

```
: RandomForestClassifier()
```

```
: rf.score(X_test, y_test)
```

```
: 0.9900166389351082
```

FIG 3.6.9: Random Forest Classifier

Classification report of the RF Classifier shows the precision recall f1 score and support metrics for each of the output labels classified by the Random Forest. Below fig represents the Classification report.

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.96	0.94	0.95	106
1	0.64	0.82	0.72	11
2	1.00	0.80	0.89	10
3	0.93	1.00	0.96	13
4	0.75	0.69	0.72	26
5	1.00	1.00	1.00	1634
6	0.75	1.00	0.86	3
accuracy			0.99	1803
macro avg	0.86	0.89	0.87	1803
weighted avg	0.99	0.99	0.99	1803

FIG 3.6.10: Classification Report of Random Forest

The Confusion matrix of this classifier against the given output classes are better depicted from the below chart

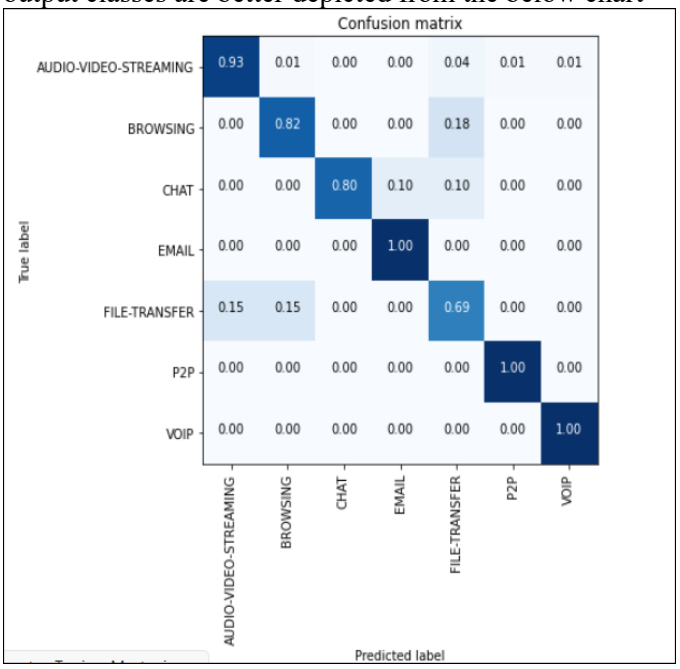

FIG 3.6.11: Confusion Matrix of Random Forest

The Feature importance depicts the role played by each feature in predicting the output classes. The graph 3.6.12 shows the feature importance for Random Forest classifier.

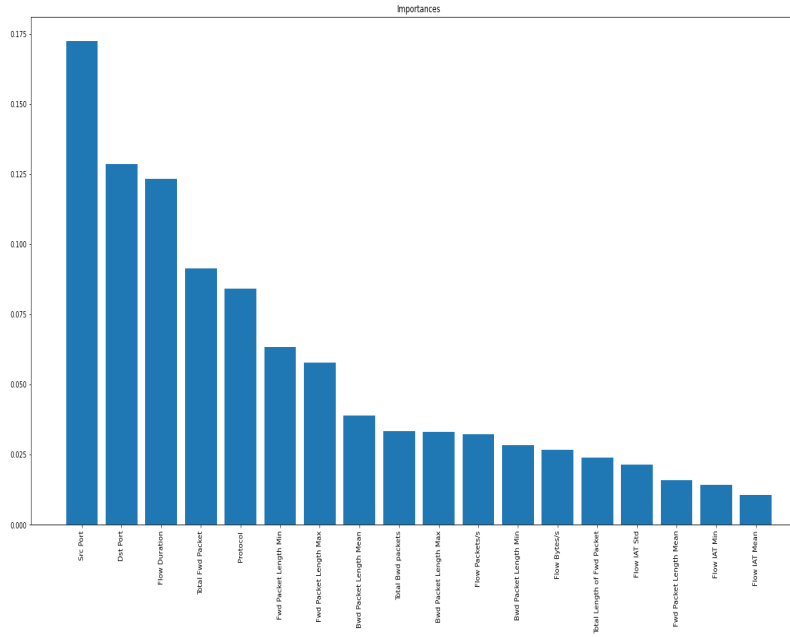

FIG 3.6.12: Feature importance of Random Forest

IV. Results and Observations.

A.Performance Evaluation

On analyzing the confusion matrices and classification reports of the all the three classifiers, it is clear that Random Forest model has the most accurate classification rate over the rest of the classifiers. With the highest overall accuracies and high classification accuracies, Random Forest was selected for this Network classification problem.

Classification model accuracies	
Model Name	**Accuracy**
Decision Tree	98.44%
K-NN Model	96.56%
Random Forest	98.90 %

TABLE 3.7: Accuracy report of classifiers

B.Further Improvements to RF-Classifier

The classification rate of Random Forest Classifier can be further improved by Hyperparameter tuning the model and selecting the best parameters to improve the model prediction rate.

On applying the below parameter list to tune the RF-mo del the accuracy rate of the classifier is improved to 98.94%.

param_grid *= {'criterion': 'gini', 'max_depth': 50, 'max_features': 'log2', 'n_estimators': 40}*

```
rf = RandomForestClassifier(max_depth=50, n_estimators=40, max_features='log2', criterion='gini')
rf.fit(X_train , y_train)
```

```
RandomForestClassifier(max_depth=50, max_features='log2', n_estimators=40)
```

```
rf.score(X_test, y_test)
```

```
0.9894620077648364
```

FIG 3.7.1: Tuned RF model

After the parameters tuned, the improved Classification rate of the Tuned RF Classifier for the output labels- '*Chat' and 'File Transfer'* classified by the Random Forest can be viewed by the below confusion matrix.

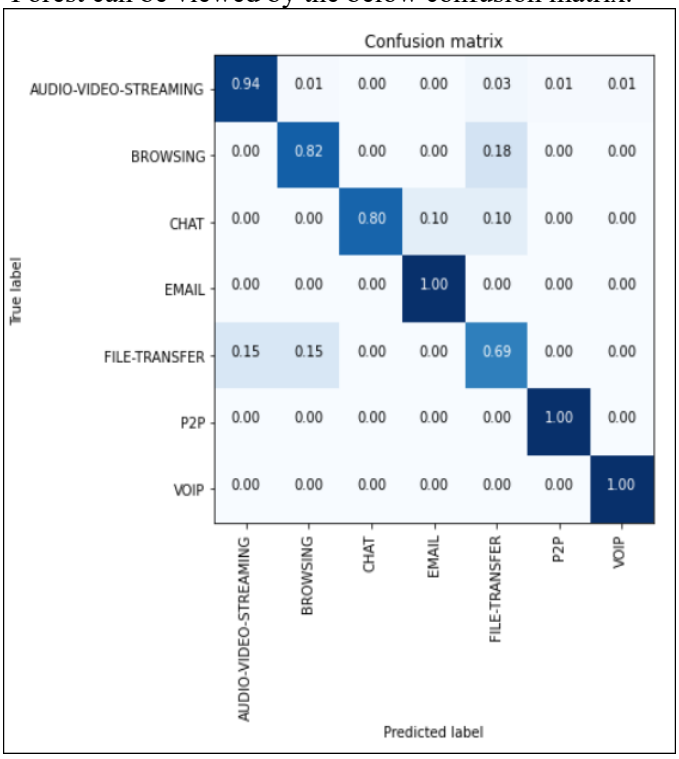

FIG 3.7.2: Confusion Matrix of Tuned RF model

CONCLUSION.

This work has been carried out as a proof of concept of combining machine learning techniques for darknet traffic classification. It can be seen that traffic classification using Supervised machine learning algorithms provides good results in classifying the Darknet traffic from both VPN and Non-VPN based networks. In the near future, these models are scaled further to readily incorporated into the network devices to classify the darknet traffic.

Also, comparative analysis of three machine learning classifiers shows the significant way of handling the classification task.

For the future work, several issues have to be addressed in terms of the classifying more output labels and better accuracy rate. Finally, for our classification problem, only three models were trained and compared.

Our attempt to train this Darknet dataset across Neural Net models RNN showed significantly less performance in classification which is less than that of above three classifiers.

However, there might be another enhanced Deep Learning based classification model that can be a better fit for this type of classification problem considering more labels and improvements with more diverse traffic labels in the real time data from Darknet.

V. ACKNOWLEDGMENT

We would like to thank Dr. Darin Johnson for his advice and guidance throughout this project, that helped to solve the ongoing research problem. Also, we are thankful for the researchers at UNB for their contribution in building CICDarknet2020 dataset.

VI. REFERENCES

[1] Waseem Iqbal Abid Khan Jadoon, Muhammad Amjad, Hammad Afzal, and Yawar Abbas Bangash. 2019. Forensic Analysis of Tor Browser: A Case Study for Privacy and Anonymity

[2] Ryoh Akiyoshi, Daisuke Kotani, and Yasuo Okabe. 2018. Detecting Emerging Large-Scale Vulnerability Scanning Activities by Correlating Low-Interaction Honeypots with Darknet. In 42nd IEEE International Conference on Computer Software & Applications. 658–663.

[3] Khaled Al-Naami, Swarup Chandra, Ahmad Mustafa, Latifur Khan, Zhiqiang Lin, Kevin Hamlen, and Bhavani Thuraisingham. 2016.Adaptive Encrypted Traffic Fingerprinting with Bi-Directional Dependence.In Proceedings of the 32nd Annual Conference on Computer Security Applications. 177–188.

[4] Mhd Wesam Al-Nabki, Eduardo Fidalgo, Enrique Alegre, and Laura Fernández-Robles. 2019. To Rank: Identifying the most influential suspicious domains in the Tor network. Expert Systems With Applications 123 (2019), 212–226.

[5] Riyad Alshammari and A. Nur Zincir-Heywood. 2008. Investigating Two Different Approaches for Encrypted Traffic Classification. In Sixth Annual Conference on Privacy, Security and Trust. 156–166

[6] Riyad Alshammari and A. Nur Zincir-Heywood. 2009. Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications. 1–8.

[7] Reyhane Attaria, Lida Abdi, and Sattar Hashemi. 2019. AdaWFPA: Adaptive Online Website Fingerprinting Attack for Tor Anonymous Network: A Stream-wise Paradigm. Computer Communications 148 (2019), 74–85.

[8] Carlos Bacquet, Kubra Gumus, Dogukan Tizer, A. Nur Zincir-Heywood, and Malcolm Heywood. 2010. A Comparison of Unsupervised Learning Techniques for Encrypted Traffic Identification. IJICR (2010), 1–9.

[9] Sikha Bagui, Xingang Fang, Ezhil Kalaimannan, Subhash C. Bagui, and Joseph Sheehan. 2017. Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features. Journal of Cyber Security Technology 1 (2) (2017), 108–126

[10] Tao Ban, Shaoning Pang, Masashi Eto, Daisuke Inoue, Koji Nakao, and Runhe Huang. 2016. Towards Early Detection of Novel Attack Patterns through the Lens of A Large-Scale Darknet. In IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress. 341–349

[11] Laurent Bernaille and Renata Teixeira. 2007. Early Recognition of Encrypted Applications. In 8th Internatinoal Conference on Passive and Active network Measurement. 165–175

[12] Laurent Bernaille, Renata Teixeira, and Kave Salamatian. 2006. Early application identification.

[13] Julian Caicedo-Muñoz, Agapito Espino, Juan Corrales, and Alvaro Rendón. 2018. QoS-Classifier for VPN and Non-VPN traffic based on time-related features. Computer Networks 144(2018), 271–279.

[14] Chen, Jennifer Tu, and Alex Vandiver. 2004. Analyzing Network Traffic from a Class B Darknet. MIT (2004)

[15] CICFlowMeter. Access February 2017. Ethernet Traffic Flow Meter. https://github.com/ahlashkari/CICFlowMeter.

[16] Fangzhou Dong, Shaoxian Yuan, Haoran Ou, and Liang Liu. 2018. New Cyber Threat Discovery from Darknet Marketplaces.

[17] Claude Fachkha, Elias Bou-Harb, and Mourad Debbabi. 2015. Inferring distributed reflection denial of service attacks from

[18] Claude Fachkha, Elias Bou-Harb, Anastasis Keliris, Nasir Memon, and Mustaque Ahamad. 2017. Internet-scale probing of cps: Inference, characterization and orchestration analysis. In Proceedings of Network and Distributed System Security Symposium 17 (2017), 100–113.

[19] Claude Fachkha and Mourad Debbabi. 2016. Darknet as a Source of Cyber Intelligence: Survey, Taxonomy, and Characterization. IEEE Communications Surveys & Tutorials 18 (2) (2016), 1197–122)

[20] Ed Wilson Tavares Ferreira and Ailton Akira Shinoda. 2016. The Development and Evaluation of a Dataset for Testing of IDS for Wireless Networks. IEEE Latin America Transactions 14, 1 (2016), 404–410.

[21] Eduardo Fidalgo, Enrique Alegre, Laura Fernández-Robles, and Víctor González-Castro. 2019. Classifying suspicious content in tor darknet through Semantic Attention Keypoint Filtering.

[22] Falguni Gadhia, Jangwon Choi, Buseung Cho, and Jungsuk Song. 2015. Comparative analysis of darknet traffic characteristics between darknet sensors. In International Conference on Advanced Communication Technology. 59–64.

[23] Gephi. [n.d.]. Gephi, the leading visualization and exploration software for all kinds of graphs and networks. https://gephi.org/. , Accessed February 2020 pages.

[24] Gerard Draper Gil, Arash H. Lashkari, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. 2016. Characterization of Encrypted and VPN Traffic Using Time-Related Features. In In Proceedings of the 2nd International Conference on Information Systems Security and Privacy. 407–414.

[25] Ramzi A. Haraty and Bassam Zantout. 2014. The TOR Data Communication System. Journal of Communications and Networks 16 (4) (2014), 415–420.

[26] Naoki Hashimoto, Seiichi Ozawa, Tao Ban, Junji Nakazato, and Jumpei Shimamura. 2018. A Darknet Traffic Analysis for IoT Malwares Using Association Rule Learning. Conference on Big Data and Deep Learning, Procedia Computer Science 144 (2018), 118–123.

[27] Gaofeng He, Ming Yang, Junzhou Luo, and Xiaodan Gu. 2014. Inferring Application Type Information from Tor Encrypted Traffic. In Second International Conference on Advanced Cloud and Big Data. 220–227.

[28] C. Rosenburg J. Early, C. Brodley. 2003. Behavioral authentication of server flows. In In Proceedings of the 19th Annual Computer Security Applications Conference.

[29] Husam Al Jawaheri, Mashael Al Sabah, Yazan Boshmaf, and Aiman Erbad. 2020. Deanonymizing Tor hidden service users through Bitcoin transactions analysis. Computers & Security 89(2020).

[30] Kota Kanemura, Kentaroh Toyoda, and Tomoaki Ohtsuki. 2019. Identification of Darknet Markets' Bitcoin Addresses by Voting Per-address Classification Results. In IEEE International Conference on Blockchain and Cryptocurrency. 154–158.

[31] Doron Kolton, Adi Stav, Asaf Wexler, Ariel Ernesto Frydman, and Yoram Zahavi. 2011. System to enable detecting attacks within encrypted traffic. United States Patent No. US 7,895,652 B2 (2011), 1–9.

[32] Yuichi Kumano, Shingo Ata, Nobuyuki Nakamura, Yoshihiro Nakahira, and Ikuo Oka. 2014. Towards Real-time Processing for Application Identification of Encrypted Traffic. In International Conference on Computing, Networking and Communications, Communication QoS and System Modeling Symposium. 136–140.

[33] Arash Habibi Lashkari, Gerard Draper-Gil, Mamun Seiful Islam, and Ali Ghorbani. 2017. Characterization of Tor Traffic Using Time Based Features. In In the proceeding of the 3rd International Conference on Information System Security and Privacy, SCITEPRESS. 253–262.

[34] Zhen Ling, Junzhou Luo, Kui Wu, Wei Yu, and Xinwen Fu. 2015. TorWard: Discovery, Blocking, and Traceback of Malicious Traffic Over Tor. IEEE Transactions on Information Forensics and Security 10 (12) (2015), 2515–2530.

[35] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Weijia Jia, and Wei Zhao. 2013. Protocol-level attacks against Tor. Computer Networks 57(2013), 869–886.

[36] Jun Liu and Kensuke Fukuda. 2014. Towards a Taxonomy of Darknet Traffic. In International Wireless Communications and Mobile Computing Conference. 37–43.

[37] Tomáš Liška, Tomáš Sochor, and Hana Sochorová. 2011. Comparison between normal and TOR-Anonymized Web Client Traffic. Procedia Computer Science 3 (2011), 888–892.

[38] Mohammad Lotfollahi, Ramin Shirali Hossein Zade, Mahdi Jafari Siavoshani, and Mohammdsadegh Saberian. 2020. Deep packet: a novel approach for encrypted traffic classification using deep learning. Soft Computing 24(2020), 1999–2012.

[39] Shane Miller, Kevin Curran, and Tom Lunney. 2018. Multilayer Perceptron Neural Network for Detection of Encrypted VPN Network Traffic. In IEEE International Conference of CNN

[40] Tomáš Minárik and Anna-Maria Osula. 2016. Tor does not stink: Use and abuse of the Tor anonymity network from the perspective of law. Computer Law & Security Review 32 (2016), 111–127.

[41] Mihnea Mirea, Victoria Wang, and Jeyong Jung. 2019. The not so dark side of the darknet: a qualitative study. Security Journal 32(2019), 102–118

[42] Antonio Montieri, Domenico Ciuonzo, Giuseppe Aceto, and Antonio Pescapé. 2018. Anonymity Services Tor, I2P, JonDonym: Classifying in the Dark (Web). IEEE Transactions on Dependable and Secure Computing (2018), 1–14.

[43] Antonio Montieri, Domenico Ciuonzo, Giampaolo Bovenzi, Valerio Persico, and Antonio Pescapé. 2019. A Dive into the Dark Web: Hierarchical Traffic Classification of Anonymity Tools. IEEE Transactions on Network Science and Engineering (2019).

[44] David Moore, Geoffrey M. Voelker, and Stefan Savage. 2006. Inferring internet denial-of-service activity. ACM Transactions on Computer Systems 24 (2) (2006), 115–139

[45] Amarnath Mullick, Shashi Nanjundaswamy, Charu Venkatraman, Junxiao He, James Harris, and Ajay Soni. 2014.

Systems and methods for application based interception of SSL/VPN traffic. Journal of Network and Computer Applications No. US 8,869,262 B2 (2014), 1–12