Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, No.12, May : 2024

DARKNET TRAFFIC ANALYSIS: ENHANCING TORNETWORK SERVICECLASSIFICATIONTHROUGHADABOOSTALGORITHM

G. Narsamma Assistant Professor, Dept. of CSE(Data Science), Sreyas Institute of Engineering and Technology, Nagole, Hyderabad
Venreddy Abhinaya UG Student, Dept. of CSE(Data Science), Sreyas Institute of Engineering and Technology, Nagole, Hyderabad
Sonaganti Sidhartha UG Student, Dept. of CSE(Data Science), Sreyas Institute of Engineering and Technology, Nagole, Hyderabad
Mohammed Neha UG Student, Dept. of CSE(Data Science), Sreyas Institute of Engineering and Technology, Nagole, Hyderabad
Tonukunuri Rishikesh UG Student, Dept. of CSE(Data Science), Sreyas Institute of Engineering and Technology, Nagole, Hyderabad

Abstract:
The Darknet, a part of the internet that is not indexed by traditional search engines, is known for its anonymity and privacy features. An essential part of the Darknet, the Tor network, is named after the onion router, which is a network of computers run by volunteers that route traffic through them to allow anonymous communication. This research investigates the use of the ensemble learning algorithm AdaBoost to improve the differentiation of Onion Service from other Tor network services in dark network traffic analysis. The network, intended for private correspondence, presents difficulties for traffic analysis because of its encryption and obfuscation. We provide a technique to increase the precision of Tor network service classification by using AdaBoost. Our methodology involves preprocessing the data, selecting features, and evaluating the model with the use of an extensive dataset. The experimental results show that compared to standard approaches, AdaBoost is effective in improving classification performance by 99%.The study has implications for enhancing network security and privacy and contributes to the progress of Darknetwork traffic analysis tools.

Index terms:
darknet, anonymity, Tor, onion services, and traffic classification

I. INTRODUCTION

The anonymity network Tor hides user identities by passing traffic across a number of middle-man nodes. Additionally, it supports Onion Services, which provides. onion domains and anonymous services. The capacity of Tor to evade censorship has caught the interest of security specialists , network defenders, and law enforcement organizations that need to differentiate Tor
traffic from other kinds. Our focus is on employing traffic analysis to separate Onion Service traffic from regular Tor traffic, whereas earlier studies tried to categorize Tor traffic or identify its application types. Unlike a conventional Tor circuit, which runs through three nodes, an Onion Service circuit passes through six nodes. Three research questions are established by us: RQ1: Recognizing traffic from Onion Service within traffic from Tor RQ2: Evaluating the effects of traffic changes, RQ3: Determining the characteristics that are important for classification.
Onion services have been used for nefarious activities, including as hosting websites that are prohibited and acting as botnet command and control servers. Governments and law enforcement organizations seek to monitor and control these services, while companies may want to limit access in order to protect their systems. Methods for recognizing traffic from Onion Services are useful for fingerprinting these services and limiting access to critical infrastructure. To mask information

leakage, Tor's traffic patterns can be changed using a variety of strategies, including padding, false bursts, delays, and traffic splitting. We can investigate RQ2 to see if changes impact Onion Service traffic's distinguishability. Although alterations affecting the validity of the categorization raise doubts about earlier studies, successful identification in spite of the revisions suggests ineffectiveness in disguising Onion Service traffic.
We concentrate on timing statistics-based characteristics and those that have been shown to be useful in exposing network traffic patterns. In order to find features that have a good correlation with different types of traffic, we use feature selection techniques. We then run trials to assess the performance of the classifier using various feature combinations.

II. LITERATUREREVIEW

Network traffic classification is crucial for tasks like traffic shaping and monitoring. Over the past 20 years, privacy- preserving technologies have grown in importance due to increased privacy concerns. Using the Tor network, which offers users privacy and facilitates anonymous services known as Onion Services, is a popular way to achieve online anonymity. This study addresses three research areas regarding the classification of onion service traffic. We assess supervised machine learning models' capacity to distinguish between traffic from Onion Service and Tor. We feed the machine learning classifiers with the fifty features we retrieved from every traffic trace.
We then investigate if contemporary Website Fingerprinting protections impact the classifiability of Tor traffic. By implementing various modifications to mask information leakage from traffic, we evaluate the influence of these protections on the classification of Onion Service traffic. Our experiments demonstrate that these classifiers, when combined with our feature set, perform worse when classed as Onion Service traffic, but they are still able to detect modified Tor traffic. We also employ three feature selection metrics—Fisher Score, Pearson's correlation, and information gain—to identify the optimal features for this assignment[1].
To choose the best features for this assignment, we also use three feature selection metrics: Fisher Score, Pearson's correlation, and information gain. When separating Tor traffic from Onion Service traffic, these top criteria successfully categorize over 98% of the traffic. On the other hand, they perform worse on altered Tor traffic traces. Traffic classification technologies have come a long way in the past ten years, particularly in areas like Quality of Service (QoS) mechanisms and SIEM (Security Information and Event Management) tools. The primary goal of this work is to use a time-based technique to detect and characterize Tor traffic. A set of characteristics that are only derived from temporal data of traffic flows involving a Tor client and a Tor entry node are utilized. The results demonstrate how well ten features are enough to identify Tor traffic. Additionally, these temporal features are able to effectively characterize Tor traffic and distinguish between various traffic applications, such as VoIP, P2P, file-transfer, and video-streaming.
In addition to providing insights into Tor detection and classification, the study illustrates the impact of flow timeout on the efficacy of the proposed approach. The classifiers perform better when flows are generated with shorter timeout values—15 seconds has been shown to be the best number. This contradicts the widely held notion that a 600-second break is suitable. The study makes the labeled dataset and the tool used to construct it freely available in order to assist other researchers in reproducing the experiment and validating their assumptions. Future work aims to expand the dataset and further explore the use of time-based variables in the detailed characterization of encrypted communication. The authors also plan to enhance their ISCXFlowMeter program[2] so that they can experiment with combining different feature sets, like flow- based and packet-based feature sets.
This article presents the design of WTF-PAD, a probabilistic link-padding protection based on adaptive padding. We compare the effectiveness and overheads of WTF-PAD with other currently

deployed link-padding-based defenses. Our research indicates that WTF-PAD provides fair protection at a reduced cost of ownership. Importantly, WTF-PAD has low bandwidth overheads and no communication delays, making it perfect for low-latency communications like Tor. We further evaluate the effectiveness of WTF-AP in open-world and multi- tab settings, demonstrating that the defense effectively reduces classifier performance to arbitrary conjecture. To make this study more efficient, we have developed a methodology that allows investigators to assess Pluggable Transports' resistance to traffic analysis. By crawling the web pluggable Transport for Tor, researchers can use this framework in the future to duplicate their traffic analysis resistance algorithms[3].

We offer novel, thin-client TrafficSliver defenses against malicious entry Onion Router (OR)-based Website Fingerprinting (WFP) attacks at the network and application layers. TrafficSliver uses user-controlled traffic splitting across many Tor channels. We evaluate the effectiveness of alternative splitting techniques that we might apply to our defenses as well. Our network-layer protection reduces attack accuracy for all state-of-the-art WFP assaults from over 98% to under 16% without introducing fake traffic or false delays.

TrafficSliver-Net significantly lowers attack accuracy whereas TrafficSliver-App accomplishes this with less client-side modifications and without necessitating changes to the underlying anonymization network. After careful analysis, we identify traffic-splitting strategies and system attributes that effectively fend off WFP attacks. Because our countermeasures are low-bandwidth, low-latency, and compatible with the current Tor network, they can be implemented in Tor[4].

Tor's user base has increased dramatically since its 2003 introduction, but the demand has not been met by the volunteer- run relays. Currently, there are over 2,500 relays worldwide. Due to the high client-to-relay ratio, users have seen subpar performance, characterized by notable and highly variable response and download delays while web browsing. We propose to use machine learning to provide tailored services in response to the need for improved Tor network performance. We find that specific QoS settings can mitigate the particular restrictions of the main traffic types on the Tor network. In order to classify our traffic records with high accuracy, we use key characteristics out of our study of traffic logs from our own network usage on the live Tor network and combine them with established classification techniques. To evaluate our approach on a real network, we also integrate our classifier into Tor and write a simple QoS rule. Our results demonstrate a high classification accuracy, which greatly improves the experience of interactive Tor users[5].

III.METHODOLOGY Proposed Research Work:

An addition to the conventional system is presented in the form of ADABOOST. By successfully classifying network traffic into Tor and Onion services with high accuracy, ADABOOST improves the overall performance of the system. When compared to the traditional system's standard machine learning techniques, it performs better. Furthermore, ADABOOST greatly increases classification accuracy, showing that it performs better in terms of accuracy and dependability than the conventional methods. The extensions project integrates the Adaboost classifier and improves Tor traffic categorization with a noteworthy 99% accuracy [14,
15], with a particular focus on onion services. With this update, traffic classification within the Tor network is now much more accurate. A user-friendly Flask framework with SQLite connectivity is created to improve practical usability, making signup and signin processes for user testing more efficient. This guarantees a smooth and safe experience, preserving user privacy and network security while making the framework usable for realistic Darknet Traffic Analysis.

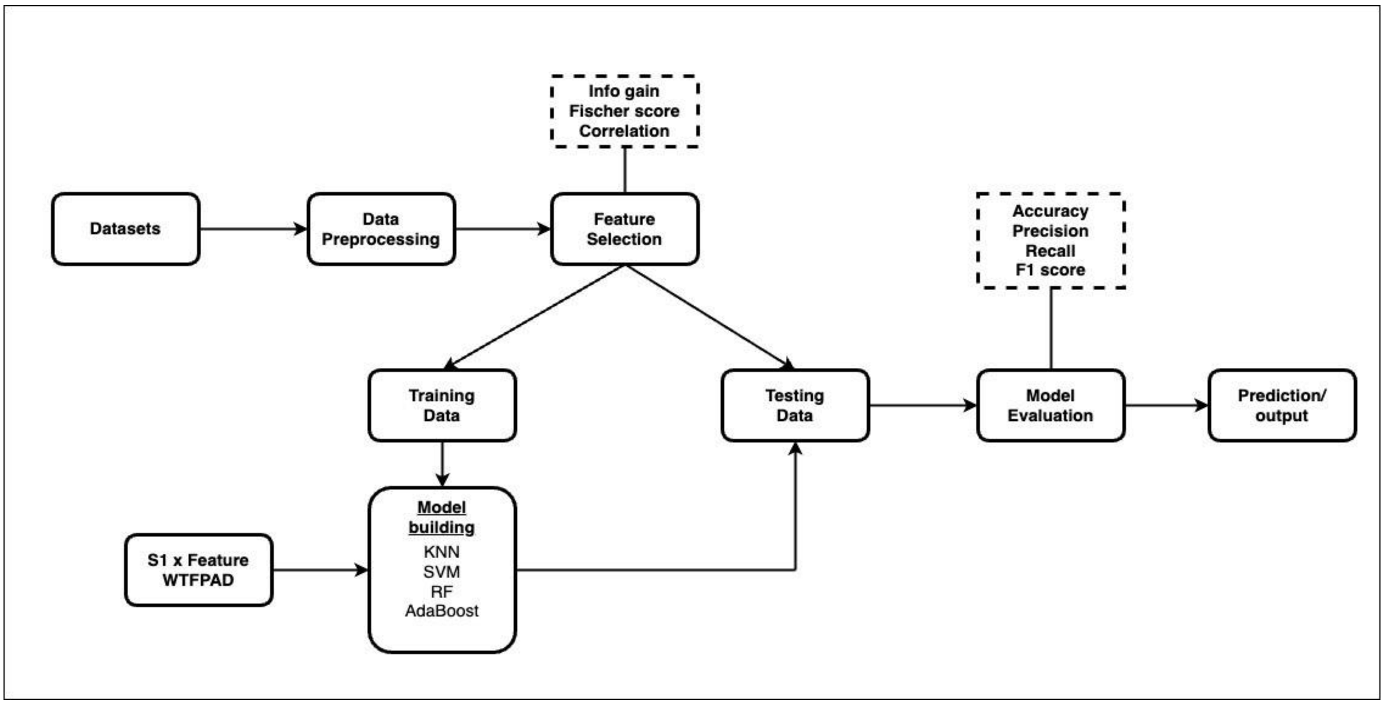


Fig 1:Block diagram of Tor Network services Classification

According to the paper's system architecture, the process is started by a Toruser. Entry Nodes (A and D) are the first points of entry into the Tor network through which the data from the Tor client enters. After passing via a B-Exit Node, the traffic leaves the Tor network and gains access to the general internet. For secure communication between clients and onion services, a C-Rendezvous Point is essential[1, 18]. The system communicates with the Tornetwork's unique Onion services as well as regular webservices. The models used to classify and comprehend the effects of changing Tor traffic on onion service traffic include Random Forest, SVM, KNN [3], and the extension of Adaboost.This extensive architecture provides vital insights into the project's objective by ensuring a complete examination into the dynamics of darknet traffic, particularly with regard to the classification of Onion services.

The algorithm for classifying services on the Tor network.

Input: Information about packet features **Output:** Tor/Onion services classification START

Step-1:Import the packages and libraries: Importing the required Python libraries and packages, including pandas, numpy, scikit-learn, matplotlib, seaborn, and others, is the first step in the code.

Step 2: Dataset Loading: The "No Defence" and "WTFPAD" datasets, together with the matching class labels, are loaded by
the code. The traffic data in the "No Defence" collection is
devoid of any defense techniques applied. The traffic data in the "WTFPAD" dataset has the WTFPAD defense mechanism applied to it.

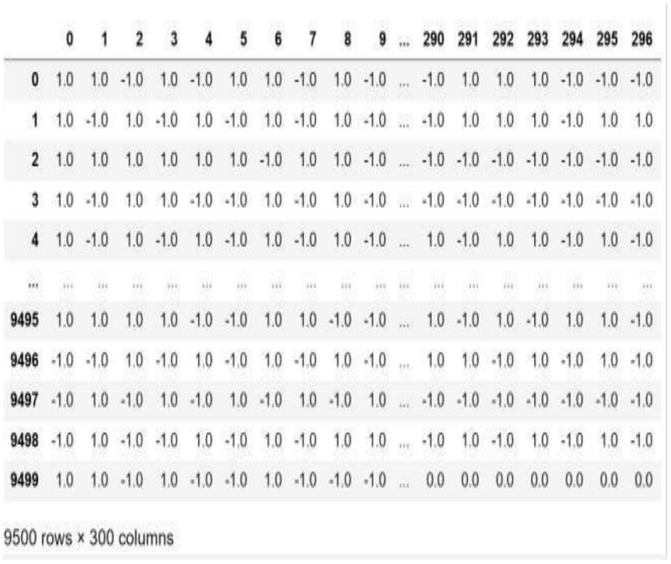
Step 3:Data Preprocessing: StandardScaler, the features are standardized by scaling them so that the variance is 1 and the mean is 0. Another name for this procedure is Z-score normalization. When features in the dataset have diverse scales, it's especially helpful.

Step 4:Feature Selection: This step involves the application of techniques like Information Gain (IG), which quantifies each feature's significance to the target variable. The highest-scoring features are chosen for model training, and they are arranged according to their IG scores.

Step 5: Data splitting: The "No Defence" and "WTFPAD" datasets are divided and mixed together. In order to assess the

effectiveness of the model, the datasets are divided into training and testing sets.
Step 6: Training Data: The machine learning model is trained using this subset of the dataset. From this data, the model picks up features, correlations, and patterns. In order to reduce the discrepancy between its predictions and the actual results in the
training data, the model iteratively modifies its parameters during the training phase.
Step 7. WTFPAD feature S1 x: The WTFPAD defensive mechanism is applied in this stage to the specific features of the dataset designated as "S1." WTFPAD is a defense method that improves Tor network traffic security and privacy. In order to
obscure traffic patterns and make it more challenging for adversaries to decipher, it adds padding to packets. **Step 8: Model Building- KNN, RF, SVM, and AdaBoost:** Using a variety of machine learning methods, including K- Nearest Neighbors (KNN), Random Forest (RF), Support Vector Machine (SVM), and AdaBoost, categorization models are constructed in this step. Since each technique has advantages and disadvantages, it can be used for a variety of datasets and classification tasks. Data points are categorized by
a. KNN according to the majority class of their closest neighbors. To arrive at the final categorization.
b. RF builds an ensemble of decision trees and aggregates their predictions.
c. SVM looks for the best hyperplane in a high-dimensional space to divide data points belonging to distinct classes.
d. AdaBoost gives misclassified data points greater weights in order to combine several weak classifiers into a strong
classifier.
Step 9: Testing Data: After the model has been trained, it must be examined to determine how well it performs with untested data. On the testing dataset, the model generates predictions, which are then contrasted with the actual results. The evaluation's performance measures aid in determining how effectively the model generalizes to fresh, untested data.
Step 10: Model evaluation: Evaluation criteria that are frequently used include F1 score, recall, accuracy, and precision .
Step 11: Prediction:whether the output will be categorized as
Tor Services or Onion Services in the end.

STOP
Dataset collection:
WTFPAD Dataset-This dataset contains network traffic modified with WTF-PAD to assess itsimpact on the classification of Tor and Onion Servicetraffic.

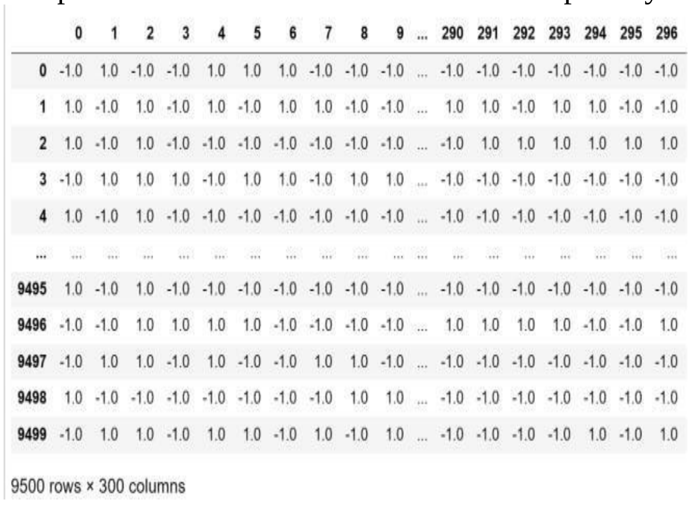


	0	1	2	3	4	5	6	7	8	9	...	290	291	292	293	294	295	296
0	1.0	1.0	-1.0	1.0	-1.0	1.0	1.0	-1.0	1.0	-1.0	...	-1.0	1.0	1.0	1.0	-1.0	-1.0	-1.0
1	1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	...	-1.0	1.0	1.0	1.0	-1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	-1.0	1.0	1.0	-1.0	...	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
3	1.0	-1.0	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	-1.0	...	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
4	1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	...	1.0	-1.0	1.0	1.0	-1.0	1.0	-1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9495	1.0	1.0	1.0	1.0	-1.0	-1.0	1.0	1.0	-1.0	-1.0	...	1.0	-1.0	1.0	-1.0	1.0	1.0	-1.0
9496	-1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	...	1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0
9497	-1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	...	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
9498	-1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	1.0	...	-1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0
9499	1.0	1.0	-1.0	1.0	-1.0	-1.0	1.0	-1.0	-1.0	-1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

9500 rows × 300 columns

Fig2:WTFPAD dataset

No Defense Dataset -This dataset represents network traffic without specific privacy defenses and serves as a baseline for comparison to evaluate the effectiveness of privacy measures in the project.



	0	1	2	3	4	5	6	7	8	9	...	290	291	292	293	294	295	296
0	-1.0	1.0	-1.0	-1.0	1.0	1.0	1.0	-1.0	-1.0	-1.0	...	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
1	1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	1.0	-1.0	-1.0	...	1.0	1.0	-1.0	1.0	1.0	-1.0	-1.0
2	1.0	-1.0	1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	...	-1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	-1.0	1.0	1.0	1.0	-1.0	1.0	1.0	-1.0	1.0	1.0	...	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
4	1.0	-1.0	1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	...	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9495	1.0	-1.0	1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	...	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
9496	-1.0	-1.0	1.0	1.0	1.0	1.0	-1.0	-1.0	-1.0	-1.0	...	1.0	1.0	1.0	1.0	-1.0	-1.0	1.0
9497	-1.0	1.0	1.0	-1.0	1.0	-1.0	-1.0	1.0	1.0	-1.0	...	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
9498	1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	1.0	1.0	...	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
9499	-1.0	1.0	1.0	-1.0	1.0	1.0	-1.0	1.0	-1.0	1.0	...	-1.0	-1.0	-1.0	-1.0	1.0	-1.0	1.0

9500 rows × 300 columns

Fig3:No Defense dataset

Data Processing

Processing data entails turning unprocessed data into useful information that companies may use. Data scientists often handle data by gathering, organizing, cleaning, verifying, analyzing, and transforming it into legible representations like documents or graphs. There are three ways to process data: mechanically, electronically, and manually. The intention is to improve information value and make decision-making easier. Businesses are able to enhance their operations and make strategic decisions on time as a result. Computer software development and other automated data processing methods are important in this. It can assist in transforming vast volumes of data, including big data, into insightful understandings for decision-making and quality control.

Features Selection

Isolating the most reliable, pertinent, and non-redundant features for use in model building is the process of feature selection. As the quantity and diversity of datasets increase, it is crucial to gradually reduce their size. Enhancing the predictive model's performance and cutting down on modeling's computational expense are the primary objectives of feature selection.

Information gain (IG): evaluates the information gain of a feature in respect to the class to determine its significance. It shows how the entropy of the class changed before and after the dataset was divided according to the feature

.

H(Class) - H(Class | Feature) equals IG (Class, Feature).

$$H(X) = -\sum_{i=1}^{N} p_i \log p_i$$

where N is the number of classes and pi is the percentage of occurrences of the ith class.

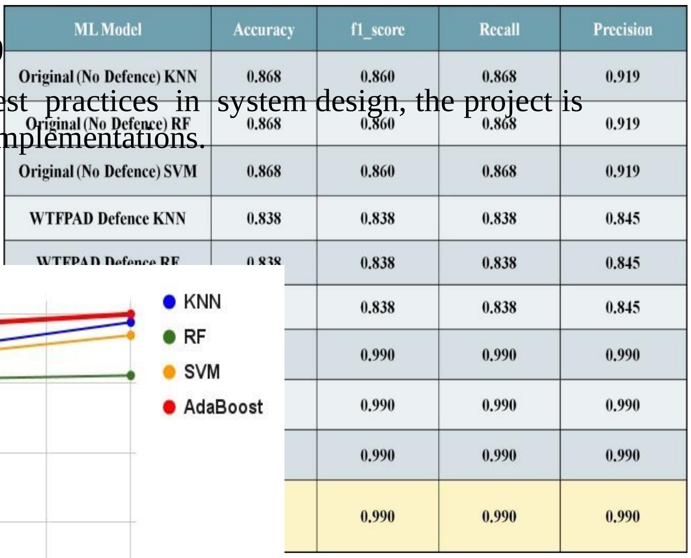
Fisher Score: The Fisher Score, which merely shows how much information a feature (like OS/TOR) may reveal about the class, is another well-liked metric in supervised learning.

$$F_k = \frac{\sum_{i=1}^{c} n_i (\bar{x}_{ki} - \bar{x}_k)^2}{\sum_{i=1}^{c} n_i (\sigma_{ki})^2}$$

where x_{ki} and σ_{ki} are the mean and variance of the kth feature in the ith Pclass, respectively; x_k is the mean of the kth feature in all classes; c is the number of classes; and n_i is the number of instances in the ith class. The fluctuation of data points inside each class is the between-class dispersion of the kth feature,
denoted by c i=1 ni(x¯ki − ¯xk) 2 in the previous equation 4. If this is high, it means that class separation is not a problem. Similarly an illustration of the within-class dispersion. A low value suggests that the data points in the class are close to one another and may be readily distinguished from one another. All things considered, a high Fisher Score means that a characteristic may be used to readily separate classes and so can be used to find the most important traits.

IV. RESULTS AND DISCUSSION

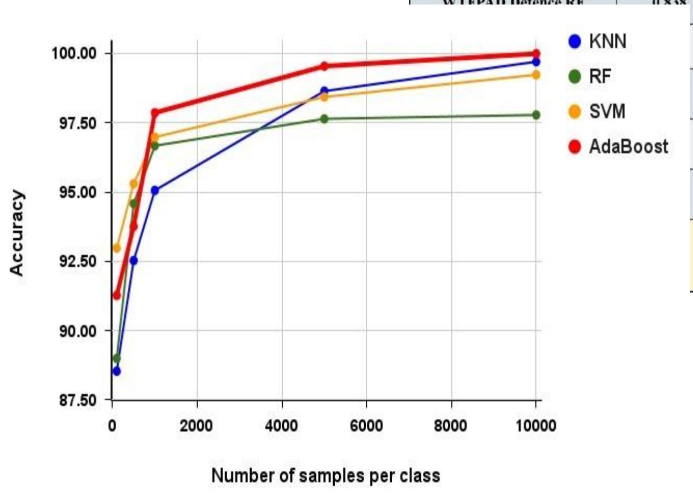
In robustness of Onion Service traffic classification under many scenarios, adding to our understanding of darknet traffic dynamics. For the classification task, the project finds and assesses the most significant feature combinations. The study clarifies the factors that have a major influence on the Onion Service traffic classification accuracy by identifying critical combinations. This information helps to improve models and gain a better comprehension of the complex interactions present in the dataset. A thorough investigation is carried out to comprehend the effectiveness of various variables and how they affect classifiers. This comprehensive analysis offers insightful information about the ways in which distinct features influence the variations in performance seen in different classifiers. The reliability and interpretability of the classification models are improved by having a better understanding of the dynamics of feature performance. By utilizing ensemble techniques—more specifically, Adaboost—the project increases its capacity and achieves higher classification accuracy for Onion Service traffic. Through the amalgamation of predictions from various distinct models, Adaboost augments the classification system's overall resilience and precision.
This work makes a valuable contribution to the delivery of an accurate and sophisticated darknet traffic analysis solution. The project combines a secure authentication system with an easy-to-use Flask interface to enhance the overall user experience during system testing. This interface guarantees a secure environment for data input for performance evaluation while streamlining user interactions.



ML Model	Accuracy	f1_score	Recall	Precision
Original (No Defence) KNN	0.868	0.860	0.868	0.919
Original (No Defence) RF	0.868	0.860	0.868	0.919
Original (No Defence) SVM	0.868	0.860	0.868	0.919
WTFPAD Defence KNN	0.838	0.838	0.838	0.845
WTFPAD Defence RF	0.838	0.838	0.838	0.845
[illegible]	[illegible]	0.838	0.838	0.845
[illegible]	[illegible]	0.990	0.990	0.990
[illegible]	[illegible]	0.990	0.990	0.990
[illegible]	[illegible]	0.990	0.990	0.990
[illegible]	[illegible]	0.990	0.990	0.990

Since security and usability go hand in hand with best practices in system design, the project is both feasible and accessible for testing and real-world implementations.

Performance Evaluation table of Tor Network Services classification.



Accuracy Comparison graph of Tor Network Services Classification

V. REFERENCES

[1] "Darknet Traffic Analysis: Investigating the Impact of Modified Tor Traffic on Onion Service Traffic Classification" Ishan Karunanayake, Nadeem Ahmed , Robert Malaney , Rafiqul Islam And Sanjay K. Jha
,2023.
[2] Juarez, Marc & Imani, Mohsen & Perry, Mike & Diaz, Claudia & Wright, Matthew. (2015), "WTF-PAD: Toward an Efficient Website Fingerprinting Defense for Tor".
[3] Cadena, Wladimir &Mitseva, Asya & Hiller, Jens & Pennekamp, Jan & Reuter, Sebastian & Filter, Julian & Engel, Thomas & Wehrle, Klaus & Panchenko, Andriy. (2020). TrafficSliver: Fighting Website Fingerprinting Attacks with Traffic Splitting. 1971-1985. 10.1145/3372297.3423351
[4] A.H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. zGhorbani, ''Characterization of Tor traffic using time based features,'' in Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy (ICISSP), Porto, Portugal, Feb. 2017, pp. 253–
262.
[5] AlSabah, Mashael & Bauer, Kevin & Goldberg, Ian. (2012). Enhancing Tor's performance using real-time traffic classification. Proceedings of the ACM Conference on Computer and Communications Security
[6] M. Kim and A. Anpalagan, ''Tor traffic classification from raw packet header using convolutional neural network,'' in Proc. 1st IEEE Int. Conf. Knowl. Innov.Invention (ICKII), Jeju
[7] G. He, M. Yang, J. Luo, and X. Gu, ''Inferring application type information from Tor encrypted traffic, Washington, DC, USA, Nov. 2014, pp. 220–227.
[8] Montieri, D. Ciuonzo, G. Aceto, and A. Pescapé,''Anonymity services tor, I2P, JonDonym: Classifying inthe dark (web),'' IEEE Trans. Dependable SecureComput., vol. 17, no. 3, pp. 662–675, May 2020.
[9] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright,''Toward an efficient website fingerprinting defense,'' inProc. 21st Eur. Symp. Res. Comput. Secur. (ESORICS), Heraklion, Greece, Sep. 2016, pp. 27–46.

[10] T. Wang and I. Goldberg, ‘‘Walkie-talkie: An efficientdefense against passive website 55 fingerprinting attacks,’’ in Proc. 26th USENIX Secur. Symp. (SEC), Vancouver, BC, Canada, Aug. 2017, pp.1375–1390.
[11] W. De la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S.Reuter, J. Filter, T. Engel, K. Wehrle, and A. Panchenko,‘‘TrafficSliver: Fighting website fingerprinting attacks with traffic splitting,’’ in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS), New York, NY, USA, Nov. 2020, pp. 1971–1985.
[12] J. Hayes and G. Danezis, ‘‘k-fingerprinting: A robust scalable website fingerprinting technique,’’ in Proc. 25th USENIX Conf. Secur. Symp. (SEC), Austin, TX, USA, Aug. 2016, pp. 1187–1203.
[13] X. Bai, Y. Zhang, and X. Niu, ‘‘Traffic identification of Tor and webmix,’’ in Proc. 8th Int. Conf. Intell. Syst. Design Appl. (ISDA), Kaohsiung, Taiwan, vol. 1, Nov.
2008, pp. 548– 551.
[14] O. Berthold, H. Federrath, and S. Köpsell, ‘‘Web MIXes: A system for anonymous and unobservable Internet access,’’ in Proc. Int. Workshop Design Issues Anonymity Unobservability, in Lecture Notes in Computer Science, vol. 2009, H. Federrath, Ed., Berkeley, CA, USA, Jul.
2000, pp. 115–129.
[15] B. Zantout and R. Haraty, ‘‘I2P data communication system,’’ in Proc. 10th Int. Conf. Netw. (ICN), Sint Maarten, The Netherlands, Jan. 2011, pp. 401–409.
[16] P. Sirinam, M. Imani, M. Juarez, and M. Wright, ‘‘Deep fingerprinting: Undermining website fingerprinting defenses with deep learning,’’ in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS), Toronto, ON, Canada, Oct. 2018, pp. 1928–1943.
[17] R. Overdorf, M. Juárez, G. Acar, R. Greenstadt, and C.
Díaz, ‘‘How unique is your.onion?: An analysis of the
fingerprintability of Tor onion services,’’ in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS), Dallas, TX, USA, Oct. 2017, pp. 2021–2036.
[18] H. Witten, E. Frank, and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2011. 56
[19] X. He, D. Cai, and P. Niyogi, ‘‘Laplacian score for feature selection,’’ in Proc. Adv. Neural Inf. Process. Syst. (NIPS),
Vancouver, BC, Canada, Dec. 2005, pp. 507–514.
[20] M. Gan and L. Zhang, ‘‘Iteratively local Fisher score for feature selection,’’ Appl. Intell., vol. 51, pp. 6167–6181, Aug. 2021