IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 14, Issue 4, April 2024, ||Series -2|| PP 51-60

An investigation of dark net traffic to see how updated tor traffic effects onion service traffic categorization

[1.] **Ippili Sravya**, *B.Tech, Department of CSE, DNR COLLEGE OF ENGINEERING AND TECHNOLOGY,Bhimavaram, Andhra Pradesh, India, ippilisravya35@gmail.com*
[2.] **Kanna Teja**, *B.Tech, Department of CSE, DNR COLLEGE OF ENGINEERING AND TECHNOLOGY,Bhimavaram, Andhra Pradesh, India, tejakanna949@gmail.com*
[3.] **Battina Gopi Chand**, *B.Tech, Department of CSE, DNR COLLEGE OF ENGINEERING AND TECHNOLOGY, Bhimavaram, Andhra Pradesh, India,battinagopichand0@gmail.com*
[4.] **Garuvu Pavan**, *B.Tech, Department of CSE, DNR COLLEGE OF ENGINEERING AND TECHNOLOGY,Bhimavaram, Andhra Pradesh, India, pavanhcl22@gmail.com*
[5.] **Dr. A Rama Murthy**, *M.Tech, PhD.,Professor, Department of CSE, DNR COLLEGE OF ENGINEERING AND TECHNOLOGY, Bhimavaram, Andhra Pradesh, India, ram111.sai@gmail.com*

Abstract:*The project centers on analyzing network traffic within darknet environments, such as the Tor network, to understand how modifications to Tor traffic impact the classification of Onion Service traffic. It acknowledges that while Tor and Onion Services are designed for privacy and anonymity, they can be misused, emphasizing the need for better understanding and monitoring. The project has three primary goals: identifying Onion Service traffic within Tor traffic, assessing the effects of traffic modifications, and pinpointing influential features in the classification process. The project likely employs machine learning and data analysis techniques to achieve its objectives, with a particular focus on analyzing network traffic patterns within the Tor network. The project's findings may have implications for privacy, security, and network monitoring, highlighting the delicate balance between maintaining user privacy and ensuring network security.*
Index terms -*Traffic classification, machine learning, onion services, tor, anonymity, feature selection.*

I. INTRODUCTION

Tor [1] is an anonymity network that hides the identity of its users by routing the traffic through multiple intermediary nodes. Tor also supports the provision of anonymous services known as Onion Services (also known as hidden services) with .onion as the top-level domain name. Tor's ability to act as a censorship circumvention tool has encouraged security experts, network defenders, and law enforcement agencies to identify Tor traffic from other encrypted and non-encrypted traffic [2], [3]. For example, [3], [4] tried to classify Tor traffic from non-Tor Traffic, [2], [5] tried to classify the applicationtypes in Tor traffic, and [6] tried to classify Tor traffic from other anonymity network traffic such as I2P traffic and Webmix Traffic. However, in this work, we intend to explore the distinguishability of Onion Service traffic from standard Tor traffic using traffic analysis. We formulate three research questions to act as a foundation for our work.

Onion Services have been used to host illegal websites, and more recently, they have been used as Command and Control (C&C) servers for botnets [7], [8]. Therefore, from the perspective of governments and law enforcement agencies, they want to track and shut down such services and regulate the Onion Service traffic [9]. Even businesses might find it useful to restrict access to such websites in order to protect their systems from potential bad actors (e.g. hackers) and attacks. As a result, having techniques for identifying Onion Service traffic can be useful for two main reasons; 1. Such techniques can act as a stepping stones for fingerprinting of Onion Services. 2. They can be useful to restrict Onion Service traffic in sensitive and confidential systems.

There are certain techniques that can be implemented in Tor to change its traffic patterns. Introducing padding [10], using dummy bursts and delays [11], and splitting the traffic [12] are a few examples of such techniques. These techniques1 have been developed with the intention of obfuscating the information leakage of Tor traffic. The main importance of answering RQ2 is that we can confirm whether our findings from RQ1 will hold true as and when such modifications are introduced to the Tor traffic. If we are able to still distinguish Onion Service traffic, it is an indication that these modifications are not effective in masking Onion Service traffic, if they are realised in the future. If the modifications do affect the Onion Service classifiability, it opens up questions about the validity of prior works, such as [3] and [6] in a setting with those modifications implemented.

Therefore, we focused on features that are focused on timing statistics. (ii) Also, we use features that have a proven track record of working well in revealing patterns in network traffic [13]. However, we use three

feature selection techniques to infer which features have a better relationship with the traffic types used in our work and conduct experiments to evaluate the classifier performance with different feature combinations.

II. LITERATURE SURVEY

We present Tor, a circuit-based low-latency anonymous communication service. This second-generation Onion Routing system addresses limitations in the original design by adding perfect forward secrecy, congestion control, directory servers, integrity checking, configurable exit policies, and a practical design for location-hidden services via rendezvous points [1]. Tor works on the real-world Internet, requires no special privileges or kernel modifications, requires little synchronization or coordination between nodes, and provides a reasonable tradeoff between anonymity, usability, and efficiency. We briefly describe our experiences with an international network of more than 30 nodes. We close with a list of open problems in anonymous communication.

Tor is a low-latency anonymity-preserving network that enables its users to protect their privacy online. It consists of volunteeroperated routers from all around the world that serve hundreds of thousands of users every day. Due to congestion and a low relayto-client ratio, Tor suffers from performance issues that can potentially discourage its wider adoption, and result in an overall weaker anonymity to all users. We seek to improve the performance of Tor by defining different classes of service for its traffic. [2] We recognize that although the majority of Tor traffic is interactive web browsing, a relatively small amount of bulk downloading consumes an unfair amount of Tor's scarce bandwidth. Furthermore, these traffic classes have different time and bandwidth constraints; therefore, they should not be given the same Quality of Service (QoS), which Tor offers them today. We propose and evaluate DiffTor, a machine-learning-based approach that classifies Tor's encrypted circuits by application in real time and subsequently assigns distinct classes of service to each application. Our experiments confirm that we are able to classify circuits we generated on the live Tor network with an extremely high accuracy that exceeds 95% [13]. We show that our real-time classification in combination with QoS can considerably improve the experience of Tor clients, as our simple techniques result in a 75% improvement in responsiveness and an 86% reduction in download times at the median for interactive users.

Traffic classification has been the topic of many research efforts, but the quick evolution of Internet servicesand the pervasive use of encryption makes it an open challenge [12, 14]. Encryption is essential in protecting theprivacy of Internet users, a key technology used in the different privacy enhancing tools that have appeared inthe recent years. Tor is one of the most popular of them, it decouples the sender from the receiver by encryptingthe traffic between them, and routing it through a distributed network of servers. In this paper [3], we present atime analysis on Tor traffic flows, captured between the client and the entry node. We define two scenarios,one to detect Tor traffic flows and the other to detect the application type: Browsing, Chat, Streaming, Mail,Voip, P2P or File Transfer. In addition, with this paper we publish the Tor labelled dataset we generated andused to test our classifiers.

As the amount of network traffic [13] is growing exponentially, traffic analysis and classification are playing a significant role for efficient resource allocation and network management. However, with emerging security technologies, this work is becoming more difficult by encrypted communication such as Tor, which is one of the most popular encryption techniques. [4] This paper proposes an approach to classify Tor traffic using hexadecimal raw packet header and convolutional neural network model [3]. Comparing with competitive machine learning algorithms, our approach shows a remarkable accuracy. To validate this method publicly, we use UNB-CIC Tor network traffic dataset. Based on the experiments, our approach shows 99.3% accuracy for the fractionized Tor/non-Tor traffic classification.

Tor is a famous anonymity communication system for preserving users' online privacy. It supports TCP applications and packs application data into encrypted equal-sized cells to hide some private information of users, such as the running application type (Web, P2P, FTP, Others). The known of application types is harmful because they can be used to reduce the anonymity set and facilitate other attacks. However, unfortunately, the current Tor design cannot conceal certain application behaviors [5]. For example, P2P applications usually upload and download files simultaneously and this behavioral feature is also kept in Tor traffic. Motivated by this observation, we investigate a new attack against Tor, traffic classification attack, which can recognize application types from Tor traffic. An attacker first carefully selects some flow features, e.g., burst volumes and directions to represent the application behaviors and takes advantage of some efficient machine learning algorithm to model different types of applications. Then these established models can be used to classify target's Tor traffic and infer its application type. We have implemented the traffic classification attack on Tor and our experiments validate the feasibility and effectiveness of the attack.

III. METHODOLOGY

i) Proposed Work:

ADABOOST is introduced as an extension to the traditional system. ADABOOST enhances the system's performance by achieving high accuracy in classifying network traffic into Tor and Onion services. It demonstrates superior performance compared to the conventional machine learning methods used in the traditional system. Additionally, ADABOOST significantly improves the accuracy of classification, indicating that it outperforms the traditional approaches in terms of accuracy and reliability.In the project titled extensions involve the integration of the Adaboost classifier, achieving a notable 99% accuracy in enhancing the classification of Tor Traffic [14,15], specifically focusing on Onion Services. This addition significantly improves the accuracy of traffic classification within the Tor network. To enhance practical usability, a user-friendly Flask framework with SQLite integration is implemented, streamlining signup and signin processes for user testing. This ensures a seamless and secure experience, making the framework accessible for practical Darknet Traffic Analysis while maintaining user anonymity and network security.

ii) System Architecture:

The system architecture of the projectinvolves a Tor user initiating the process. The Tor client's data enters through Entry Nodes (A and D), serving as the initial points of entry into the Tor network. The traffic then passes through a B-Exit Node, exiting the Tor network and accessing the regular internet. For Onion services, a C-Rendezvous Point is crucial for secure communication between clients and Onion services [1, 18]. The system interacts with both normal web services and special Onion services within the Tor network. The models employed, including KNN [3], Random Forest, SVM, and the extension of Adaboost, play a pivotal role in classifying and understanding the impact of modified Tor traffic on Onion Service Traffic. This comprehensive architecture ensures a thorough investigation into the dynamics of darknet traffic, especially concerning the classification of Onion services, contributing valuable insights to the project's objectives.

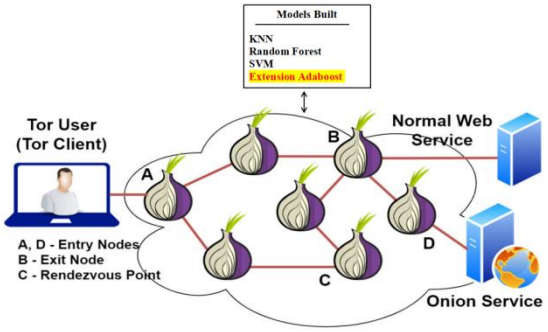


Fig 1 Proposed architecture

iii) Dataset collection:

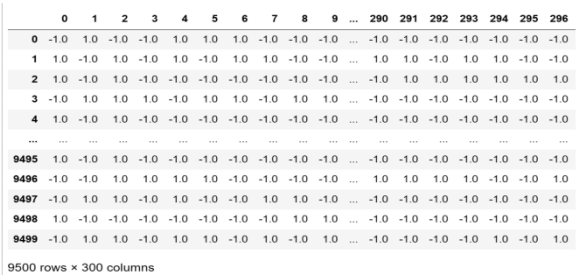
WTFPAD [10] Dataset -This dataset contains network traffic modified with WTF-PAD to assess its impact on the classification of Tor and Onion Service traffic.

	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

Fig 2 WTFPAD dataset

No Defence 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

Fig 3 No Defense dataset

iv) Data Processing:

Data processing involves transforming raw data into valuable information for businesses. Generally, data scientists process data, which includes collecting, organizing, cleaning, verifying, analyzing, and converting it into readable formats such as graphs or documents. Data processing can be done using three methods i.e., manual, mechanical, and electronic. The aim is to increase the value of information and facilitate decision-making. This enables businesses to improve their operations and make timely strategic decisions. Automated data processing solutions, such as computer software programming, play a significant role in this. It can help turn large amounts of data, including big data, into meaningful insights for quality management and decision-making.

v) Feature selection:

Feature selection is the process of isolating the most consistent, non-redundant, and relevant features to use in model construction. Methodically reducing the size of datasets is important as the size and variety of datasets continue to grow. The main goal of feature selection is to improve the performance of a predictive model and reduce the computational cost of modeling.

Feature selection, one of the main components of feature engineering, is the process of selecting the most important features to input in machine learning algorithms. Feature selection techniques are employed to reduce the number of input variables by eliminating redundant or irrelevant features and narrowing down the set of features to those most relevant to the machine learning model. The main benefits of performing feature selection in advance, rather than letting the machine learning model figure out which features are most important.

vi) Algorithms:

K-Nearest Neighbors (KNN) is a straightforward classification algorithm that categorizes data points by comparing their similarity to nearby data points. It uses features from network traffic to determine "closeness" between samples and assigns them to the most common class among their k-nearest neighbors. KNN is easy to implement and can handle complex data relationships, but challenges include selecting the appropriate value of 'k' and managing high-dimensional data [13].

```
#train KNN algorithm on No-Defence Dataset
#defining KNN tuning parameters
tuning_param = {'n_neighbors' : [1], 'p' : [1]}
knn_no_defence = GridSearchCV(KNeighborsClassifier(), tuning_param, cv=
start = timeit.default_timer()
knn_no_defence.fit(no_def_X, no_def_Y)#now train KNN
end = timeit.default_timer()
predict = knn_no_defence.predict(def_X_test) #perfrom prediction on tes
end1 = timeit.default_timer()
calculateMetrics("Original (No Defence) KNN", predict, def_y_test, (enc
```

Fig 4 KNN

Random Forest is an ensemble learning method that blends multiple decision trees for predictions. Each tree is trained on random data subsets with replacement (bagging), and the final prediction relies on votes from all the trees. It's suitable for classifying network traffic as it combines different trees to enhance accuracy. It's robust, handles high-dimensional data, and reduces overfitting, offering feature importance insights.

```
#train Random Forest algorithm on No-Defence Dataset
#defining Random Forest tuning parameters
tuning_param = {'n_estimators' : [90]}
rf_no_defence = GridSearchCV(RandomForestClassifier(), tuning_param, cv
start = timeit.default_timer()
rf_no_defence.fit(no_def_X, no_def_Y)#now train KNN
end = timeit.default_timer()
predict = rf_no_defence.predict(def_X_test) #perfrom prediction on test
end1 = timeit.default_timer()
calculateMetrics("Original (No Defence) Random Forest", predict, def_y_
```

Fig 5 Random forest

Support Vector Machines (SVM) [3] is a supervised learning algorithm that seeks an optimal hyperplane in high-dimensional space to effectively separate different data classes, excelling in cases with distinct class boundaries. It's applicable for classifying network traffic, particularly in binary or multi-class scenarios. SVM handles high-dimensional data, defines a clear margin, but may struggle with non-linearly separable data and requires careful selection of the kernel function.

```
#train SVM algorithm on No-Defence Dataset
#defining SVM tuning parameters
tuning_param = {'C' : [100], 'kernel': ['rbf']}
svm_no_defence = GridSearchCV(svm.SVC(), tuning_param, cv=5)#defining s
start = timeit.default_timer()
svm_no_defence.fit(no_def_X, no_def_Y)#now train KNN
end = timeit.default_timer()
predict = svm_no_defence.predict(def_X_test) #perfrom prediction on tes
end1 = timeit.default_timer()
calculateMetrics("Original (No Defence) SVM", predict, def_y_test, (enc
```

Fig 6 SVM

AdaBoost (Adaptive Boosting) is an ensemble machine learning algorithm that combines the predictions of multiple weak learners to create a strong and accurate classifier. It works by giving more weight to misclassified data points in each iteration, allowing weak learners to focus on the previously misclassified samples.

```
#extension XGBoost training on merge dataset
ab_cls = AdaBoostClassifier(n_estimators=100)
start = timeit.default_timer()
ab_cls.fit(X_train, y_train)#now train KNN
end = timeit.default_timer()
predict = ab_cls.predict(X_test) #perfrom prediction on test data
end1 = timeit.default_timer()
calculateMetric("WTFPAD Extension AdaBoost Top 6 Features", predict, y_
```

Fig 6 AdaBoost

IV. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:
Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

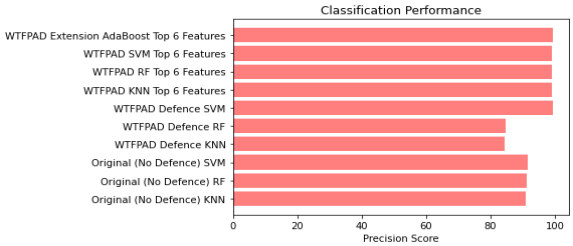


Fig 7 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$Recall = \frac{TP}{TP + FN}$$

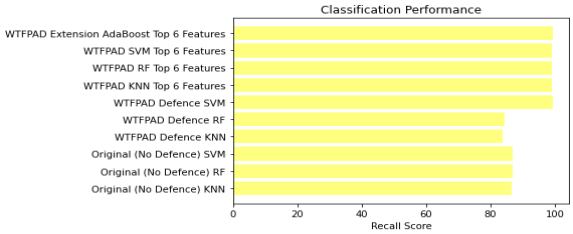


Fig 8 Recall comparison graph

Accuracy: Accuracy is the proportion of correct predictions in a classification task, measuring the overall correctness of a model's predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

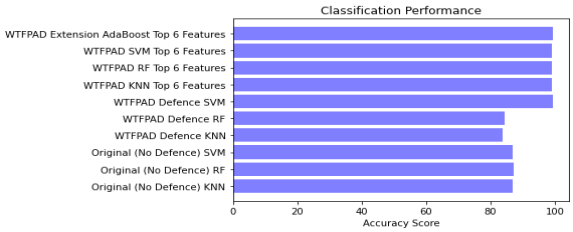


Fig 9 Accuracy graph

F1 Score: The F1 Score is the harmonic mean of precision and recall, offering a balanced measure that considers both false positives and false negatives, making it suitable for imbalanced datasets.

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$

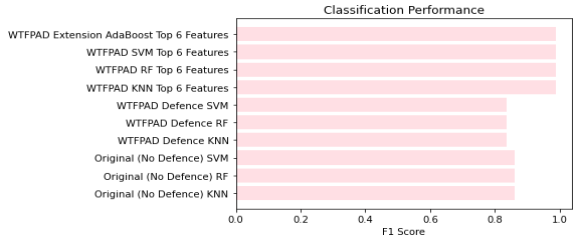


Fig 10 F1Score

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
WTFPAD Defence SVM	0.838	0.838	0.838	0.845
WTFPAD KNN Top 6 Features	0.990	0.990	0.990	0.990
WTFPAD RF Top 6 Features	0.990	0.990	0.990	0.990
WTFPAD SVM Top 6 Features	0.990	0.990	0.990	0.990
Extension WTFPAD Extension AdaBoost Top 6 Features	0.990	0.990	0.990	0.990

Fig 11 Performance Evaluation

Fig 12 Home page

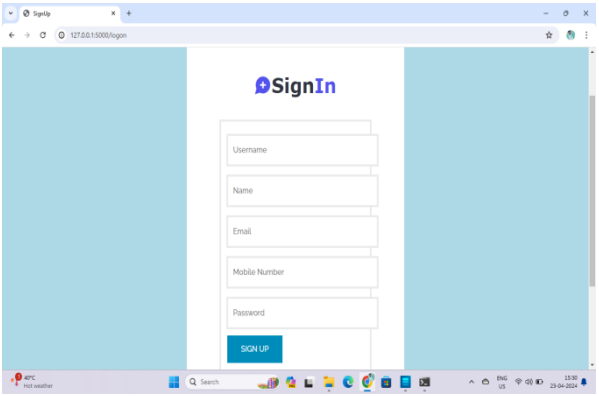


Fig 13 Signin page

Fig 14 Login page

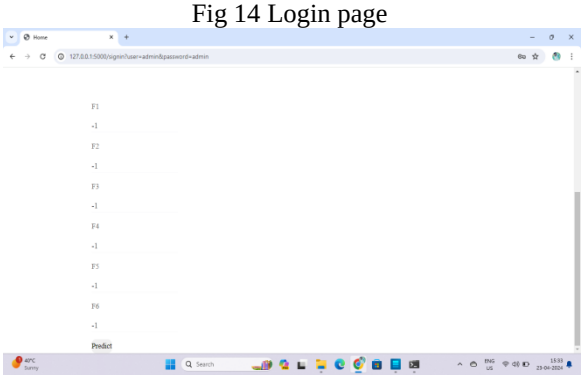


Fig 15User input 1

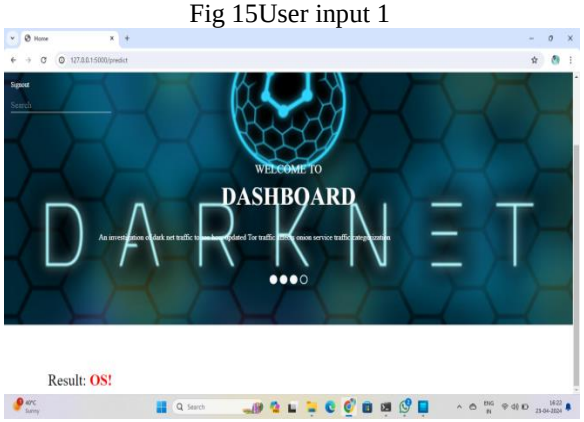


Fig 16 Predict result for given input 1

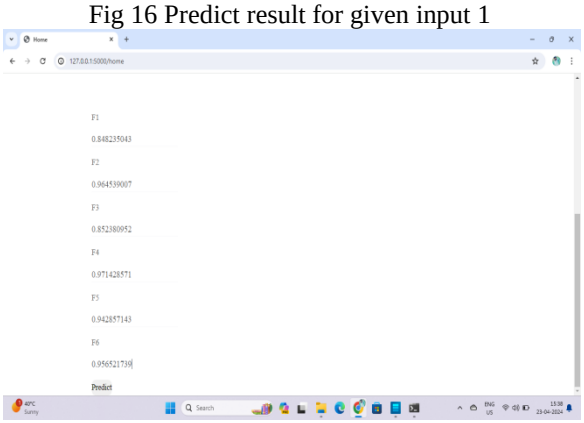


Fig 17 User Input 2

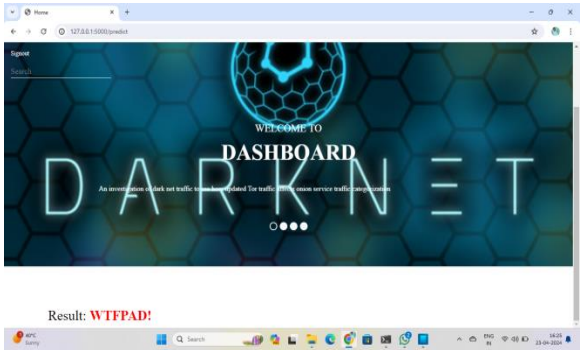


Fig 18 Predict Result For User Input 2

V. CONCLUSION

The project successfully delves into the impact of modified Tor traffic on the classification of Onion Service traffic. Through rigorous analysis, it uncovers that modifications to Tor traffic have a discernible effect on the accuracy of classification. This insight is crucial for understanding the resilience of Onion Service traffic classification under varying conditions, contributing to the broader knowledge of darknet traffic dynamics([3], [6]). The project identifies and evaluates the most influential feature combinations for the classification problem. By pinpointing key combinations, the study sheds light on the variables that significantly impact the accuracy of Onion Service traffic classification. This knowledge aids in refining models and developing a deeper understanding of the intricate relationships within the dataset. In-depth exploration is conducted to understand the performance of different features and their impact on classifiers. This thorough analysis provides valuable insights into how various features contribute to the performance changes observed in different classifiers. Understanding the dynamics of feature performance enhances the interpretability and reliability of the classification models. The project extends its capabilities by employing ensemble techniques, specifically Adaboost, to achieve heightened accuracy in Onion Service traffic classification. By combining the predictions of multiple individual models, Adaboost enhances the overall robustness and accuracy of the classification system. This project contributes to the project's goal of delivering a sophisticated and accurate solution for darknet traffic analysis. To improve the overall user experience during system testing, the project integrates a user-friendly Flask interface with secure authentication. This interface streamlines user interactions and ensures a secure environment for inputting data for performance evaluation. The combination of user-friendliness and security aligns with best practices in system design, making the project accessible and practical for testing and real-world applications.

VI. FUTURE SCOPE

To assess the impact of Tor traffic modifications, it is essential to conduct further research on classifier performance when Tor traffic is intentionally altered with tools like WTFPAD [10] or TrafficSliver. Such modifications can affect the ability to distinguish Tor from non-Tor traffic, which has implications for network monitoring and security. The project can delve into how different features affect classifier performance in the identification of Onion Service traffic. This exploration offers insights into the features' impact on the effectiveness of classification. It is crucial to conduct additional research to comprehend the consequences of easily identifying Tor and Onion Service traffic([3], [6]). This understanding should encompass traffic monitoring and potential restrictions imposed by governments and sensitive institutions, considering the broader societal and security implications. The project may explore advanced techniques to enhance the accuracy of identifying Onion Service traffic, even in the presence of obfuscation techniques. This research could lead to innovative methods for maintaining classification accuracy within the Tor network.

REFERENCES

[1]. R. Dingledine, N. Mathewson, and P. Syverson, ‘‘Tor: The secondgeneration onion router,’’ in Proc. 13th USENIX Secur. Symp. (SSYM), San Diego, CA, USA, Aug. 2004, pp. 303–320.

[2]. M. Al Sabah, K. Bauer, and I. Goldberg, ‘‘Enhancing Tor’s performance using real-time traffic classification,’’ in Proc. ACM Conf. Comput. Commun. Secur. (CCS), New York, NY, USA, Oct. 2012, pp. 73–84.

[3]. A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, ‘‘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.

[4]. 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 Island, South Korea, Jul. 2018, pp. 187–190.
[5]. G. He, M. Yang, J. Luo, and X. Gu, ‘‘Inferring application type information from Tor encrypted traffic,’’ in Proc. 2nd Int. Conf. Adv. Cloud Big Data (CBD), Washington, DC, USA, Nov. 2014, pp. 220–227.
[6]. A. Montieri, D. Ciuonzo, G. Aceto, and A. Pescapé, ‘‘Anonymity services tor, I2P, JonDonym: Classifying in the dark (web),’’ IEEE Trans. Dependable Secure Comput., vol. 17, no. 3, pp. 662–675, May 2020.
[7]. (May 2017). WCry Ransomware Analysis. Accessed: Apr. 26, 2023. [Online]. Available: https://www.secureworks.com/research/wcryransomware-analysis
[8]. (Jul. 2019). Keeping a Hidden Identity: Mirai C&Cs in Tor Network. Accessed: Apr. 26, 2023. [Online]. Available: https://blog.trendmicro. com/trendlabs-security-intelligence/keeping-a-hidden-identity-mirai-ccsin-tor-network/
[9]. (Nov. 2014). Global Action Against Dark Markets on Tor Network. Accessed: Aug. 4, 2020. [Online]. Available: https://www.europol. europa.eu/newsroom/news/global-action-against-dark-markets-tornetwork
[10]. M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, ‘‘Toward an efficient website fingerprinting defense,’’ in Proc. 21st Eur. Symp. Res. Comput. Secur. (ESORICS), Heraklion, Greece, Sep. 2016, pp. 27–46.
[11]. T. Wang and I. Goldberg, ‘‘Walkie-talkie: An efficient defense against passive website fingerprinting attacks,’’ in Proc. 26th USENIX Secur. Symp. (SEC), Vancouver, BC, Canada, Aug. 2017, pp. 1375–1390.
[12]. 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.
[13]. 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.
[14]. 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.
[15]. 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.
[16]. 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.
[17]. 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.
[18]. 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.
[19]. I. 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.
[20]. 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.
[21]. M. Gan and L. Zhang, ‘‘Iteratively local Fisher score for feature selection,’’ Appl. Intell., vol. 51, pp. 6167–6181, Aug. 2021.