



# **iJRASET**

International Journal For Research in  
Applied Science and Engineering Technology



---

# **INTERNATIONAL JOURNAL FOR RESEARCH**

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

---

**Volume: 13    Issue: III    Month of publication: March 2025**

**DOI: <https://doi.org/10.22214/ijraset.2025.67503>**

**[www.ijraset.com](http://www.ijraset.com)**

**Call:  08813907089**

**E-mail ID: [ijraset@gmail.com](mailto:ijraset@gmail.com)**

# Darknet Traffic Analysis: Investigating the Impact of Modified Tor Traffic on Onion Service Classification

Mrs. S. Srilatha<sup>1</sup>, K. Jyotsna<sup>2</sup>, B. Shiva Kumar<sup>3</sup>, V. Abhiram<sup>4</sup>, T. Jeevan Venkat Sai<sup>5</sup>

<sup>1</sup>M.Tech, Assistant Professor (Project Guide), Dept of CSE, Raghu Engineering College

<sup>2, 3, 4, 5</sup> Dept of CSE, Raghu Institute of Technology

**Abstract:** *Classifying network traffic is important for traffic shaping and monitoring. In the last two decades, with the emergence of privacy concerns, the importance of privacy-preserving technologies has risen. The Tor network, which provides anonymity to its users and supports anonymous services known as Onion Services, is a popular way to achieve online anonymity. However, this anonymity (especially with Onion Services) is frequently misused, encouraging governments and law enforcement agencies to de-anonymise them. Therefore, in this paper, we try to identify the classifiability of Onion Service traffic, focusing on three main contributions. First, we try to identify Onion Service traffic from other Tor traffic. The techniques we have used can identify Onion Service traffic with >99% accuracy. However, there are several modifications that can be done to the Tor traffic to obfuscate its information leakage. In our second contribution, we evaluate how our techniques perform when such modifications have been done to the Tor traffic. Our experimental results show that the second conditions make the Onion Service traffic less distinguishable (in some cases, the accuracy drops by more than 15%). In our final contribution, we identify the most influential feature combinations for our classification problem and evaluate their impact.*

**Index terms:** *Tor network, Onion service, network traffic analysis, privacy, anonymity, traffic classification, machine learning, security.*

## 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 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 application types in Tor traffic, and [6] tried to classify Tor traffic from other anonymity network traffic such as I2P traffic and Web mix 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. Second, we try to investigate the same problem under different settings. Specifically, we try to investigate RQ2: How do our results for RQ1 hold when we use modified Tor traffic? 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 techniques 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. As outcomes of RQ2 can open up further research avenues on Tor traffic classification, we argue RQ2 is worth evaluating. In order to investigate RQ1 and RQ2, we employ passive network analysis, in which we utilise traces of network traffic captured at points between the client and the Tor entry node. We first extract fifty features from each traffic trace, which creates a unique fingerprint of that trace. A traffic trace refers to a set of consecutive packets transmitted between the client and the entry node in a given duration. We use three machine learning classifiers that have shown promising results in network traffic classification in the past and evaluate how they work in our scenarios.

## II. LITERATURE SURVEY

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second Generation Onion Router," in Proceedings of the 13th USENIX Security Symposium, SSYM'04, (San Diego, CA, USA), pp. 303-320, August 2004. 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. 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. [2] M. AlSabah, K. Bauer, and I. Goldberg, "Enhancing Tor's Performance Using Real-Time Traffic Classification," in Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, (New York, NY, USA), pp. 73-84, October 2012. Tor is a low-latency anonymity-preserving network that enables its users to protect their privacy online. It consists of volunteer-operated routers from all around the world that serve hundreds of thousands of users every day. Due to congestion and a low relay-to-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. 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%. 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. [3] A. H. Lashkari., G. D. Gil., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of Tor Traffic using Time based Features," in Proceedings of the 3rd International Conference on Information Systems Security and Privacy, ICISSP 2017, (Porto, Portugal), pp. 253-262, February 2017. Traffic classification has been the topic of many research efforts, but the quick evolution of Internet services and the pervasive use of encryption makes it an open challenge.

Encryption is essential in protecting the privacy of Internet users, a key technology used in the different privacy enhancing tools that have appeared in the recent years.

Tor is one of the most popular of them, it decouples the sender from the receiver by encrypting the traffic between them, and routing it through a distributed network of servers. In this paper, we present a time 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 and used to test our classifiers.

## III. METHODOLOGY

### A. Proposed Work

The proposed work introduces two improved channel attention modules: a space-time (ST) interaction module, which captures detailed spatiotemporal features through matrix operations, and a depth wise separable convolution module, which processes spatial and channel information independently for more refined feature extraction. This model uses a multi-scale CNN designed to handle sequential data, applying low-rank learning to each data segment and connecting them along the time axis to form a cohesive activity representation.

To enhance cross-architecture flexibility, we introduce feature similarity functions that reduce differences in extracted features at different network levels, allowing improved transferability of the model's recognition capabilities across various network structures. Overall, the approach boosts classification accuracy while reducing computational demands, making it suitable for practical applications that need efficiency and high performance. Advantages: \* The system's high accuracy ensures reliable identification of Onion Service traffic within Tor. \* Versatile evaluation examines various scenarios, revealing privacy measure effectiveness. \* Traditional classifiers provide transparent and interpretable results.



B. System Architecture

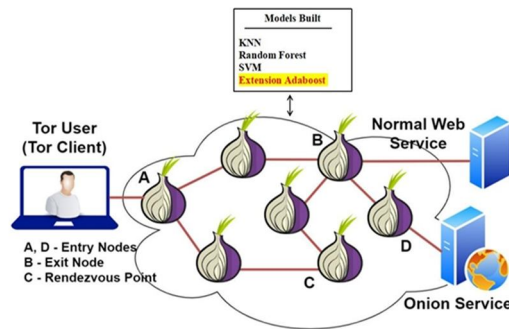


Fig.5.1.1 System architecture

Fig 1: System Architecture

C. Data Set Collection

the data in this collection is based on suspicious activities, such as cyber bullying that uses racial and ethnic profiling extensively and makes threats and uses harsh language. The information was sourced via Twitter and Face book groups. Data is categorised according to the presence or absence of potentially problematic language in tweets and comments. After data scraping, manually assign a questionable data value of -1 and a non-doubtful data value of 0. We will be using this module to get the data ready for analysis. This module allows us to create a 'train set' and a 'test set' from the information we have. The process is called model making. Naïve Bayes and logistic regression are used. The Idea of Bayes Probability and Fuzzy GA An amalgamation of LR fuzzy GA, RF, AB, and stacking classifiers into a single voting classifier. Accuracy of algorithms' predictions This module will be used to gather the login and registration information of users. The users supplied data: To gather information for forecasts, you may use this module. The projected final outcome is shown. Using an ensemble approach to combine the predictions of many models led to a more accurate and reliable final forecast. Improvements in performance, maybe as high as 99% accuracy, are possible with the addition of two additional ensemble methods: Voting Classifier and Stacking Classifier. Formulas for calculations [F] One method of classification that relies on Bayes' Theorem is Naive Bayes, which assumes that predictors are independent. A Naive Bayes classifier, in its most basic form, takes for granted that there is no correlation between two features that share a class. To solve binary classification issues, logistic regression determines the likelihood of a result, occurrence, or observation. It falls within the category of supervised machine learning. The model can only provide a yes/no, 0/1, or true/false result. The Naive Bayes algorithm is among the most well-known and user-friendly approaches to machine learning classification, alongside Naive Bayes Fuzzy GA. One may calculate probabilities and conditional probabilities using Bayes' Theorem. One definition of a fuzzy genetic algorithm is an algorithm that uses fuzzy logic techniques to build some of its components. The algorithm is thereafter described as a sequence of commands. Because they lack both volume and mass, particles serve as the basic unit of measurement in the PSO algorithm. One such example is Naive Bayes Fuzzy PSO. The trajectories of particles are fine-tuned by constantly changing their velocities in response to their own and other particles' flight experiences within the search space. that An ML model called the Voting Classifier (AB + RF) accepts a number of models as input and returns the class has the best chance of being selected. One ensemble approach to machine learning is stacking classifiers, which looks for methods to integrate the predictions of many successful ML models. Many Python programmers prefer to use the scikit-learn package when they need to build stacking ensembles.

	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 x 300 columns

Fig 2: Data set collection

#### D. Data Processing

Data processing converts unrefined facts to meaningful knowledge for organizations. Data scientists usually interact in data processing, include a collection, organization, cleaning, verification, evaluation and transformation of facts into understandable formats, including graphs or articles. records processing may be carried out thru 3 methods: “manual, mechanical, and electronic”. Here the goal is to decorate the knowledge price and simplify decisions. This allows companies to make quick strategic decisions and increase operations. automated facts processing technologies, along with laptop software program programming, are pivotal in this context. it can rework enormous volumes of facts, in particular large facts, into great insights for nice control and decision-making.

#### E. Feature Selection

Feature selection is determining for model development the maximum steady, non-redundant, and relevant capabilities. Since the quantity and variety of datasets hold to upward thrust, methodically decreasing dataset size is genuinely crucial. Characteristic preference's predominant objectives are to minimize computing cost of modeling and enhance the overall performance of a predictive model. A basic factor of characteristic engineering, feature selection is determining the greatest superb capacities for input into system gaining knowledge of algorithms. By omitting redundant or superfluous abilities, feature selection methods help to lessen the range of input variables, so enhancing the set to the ones most relevant to the system discovering version. The foremost benefits of the usage of feature preference earlier rather than allowing the device studying model to determine the relevance of features independently.

#### F. Algorithms

##### 1) “K-Nearest Neighbors (KNN)”:

It 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.

```
#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 3 KNN”

##### 2) “Random Forest”

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 4 RF”

3) “SVM”

Support Vector Machines (SVM) 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
```

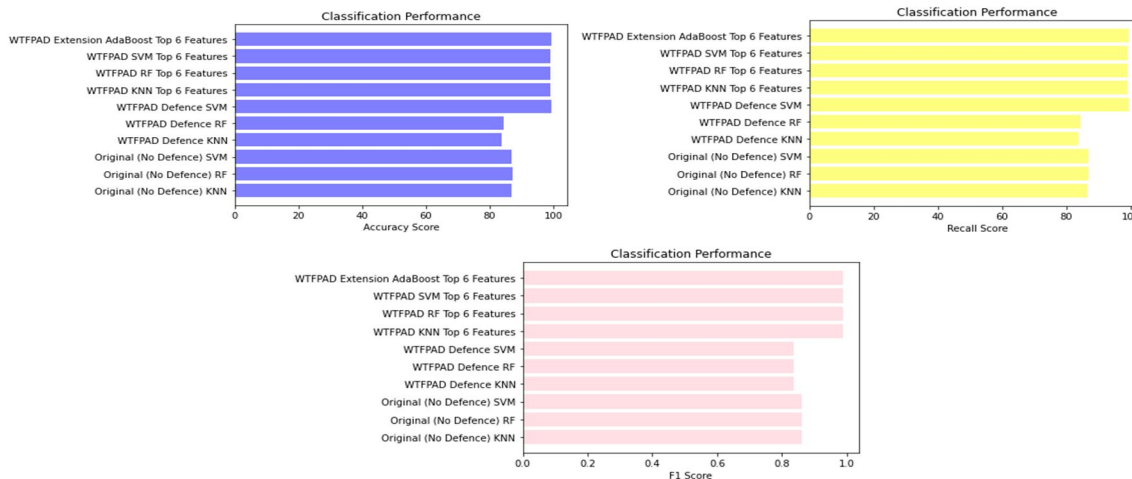
“Fig 5 SVM”

4) 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_
```

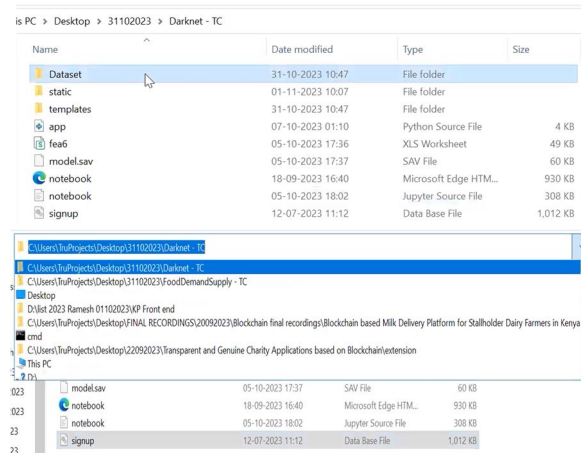
IV. CLASSIFICATION PERFORMANCE



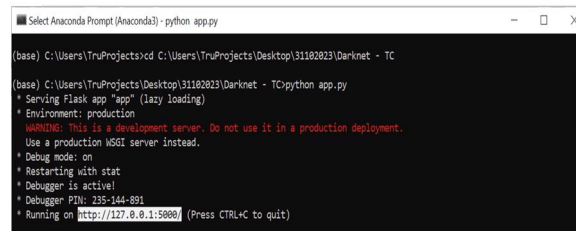
“Fig 6: Recall comparison graph”

## V. EXEXUTION

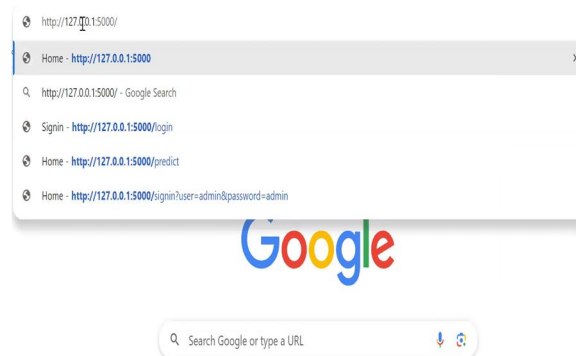
### 1) Stage 1



### 2) Stage 2



### 3) Stage 3



### 4) Stage 4

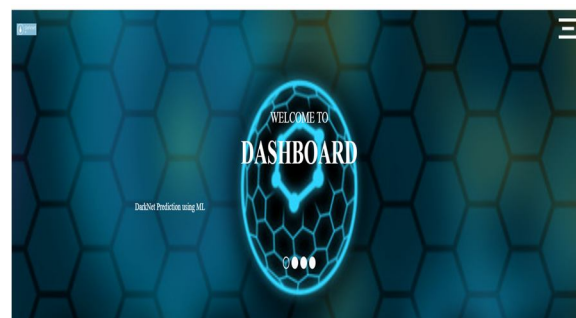
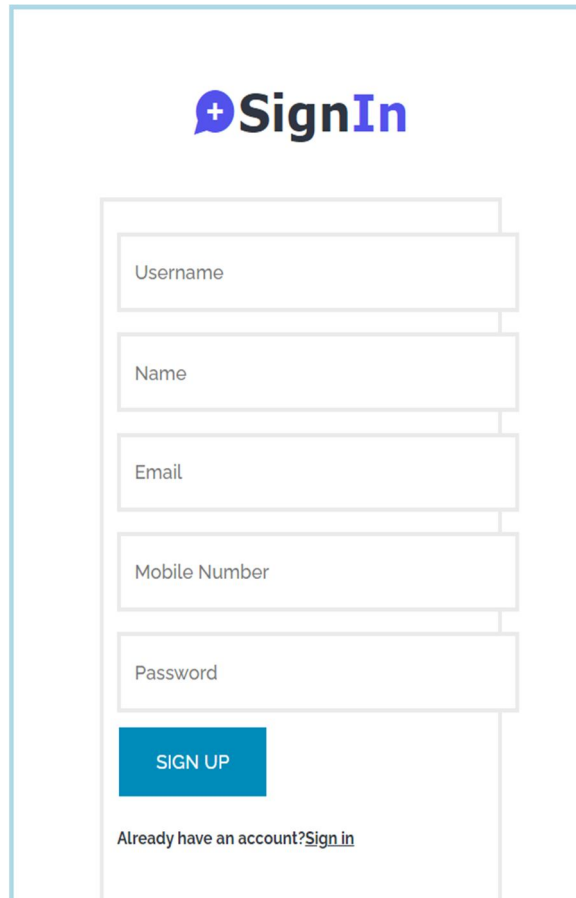


Fig 7 : Dashboard page

5) Stage 5

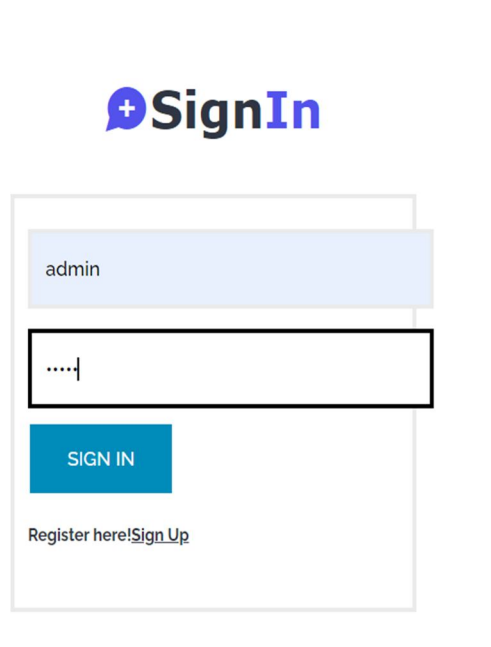


The screenshot shows a 'Sign In' page with the following elements:

- Logo: A blue circle with a white plus sign followed by the text 'Sign In' in blue.
- Form Fields: Five stacked input fields with labels: 'Username', 'Name', 'Email', 'Mobile Number', and 'Password'.
- Button: A blue button with the text 'SIGN UP' in white.
- Text: Below the button, the text 'Already have an account? [Sign in](#)' is displayed.

Fig 8: sign in page

6) Stage 6



The screenshot shows a 'Sign In' page with the following elements:

- Logo: A blue circle with a white plus sign followed by the text 'Sign In' in blue.
- Form Fields: Two input fields. The first contains the text 'admin'. The second contains a masked password '.....|' and is highlighted with a black border.
- Button: A blue button with the text 'SIGN IN' in white.
- Text: Below the button, the text 'Register here! [Sign Up](#)' is displayed.

Fig 9: login credentials





7) Stage 7

F1

---

F2

---

F3

---

F4

---

F5

---

F6

---

Predict

Fig 10: input acceptance by user

8) Stage 9

F1

-1

---

F2

-1

---

F3

-1

---

F4

-1

---

F5

-1

---

F6

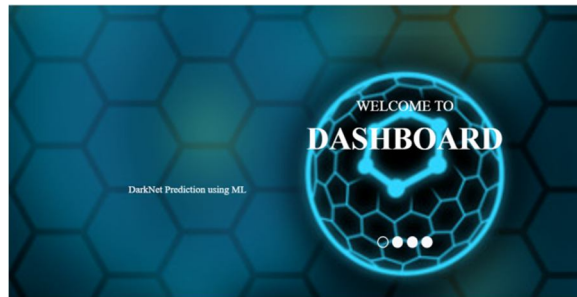
-1

---

Predict

Fig 11: output processing

9) Stage 10



Result: **OS!**

Fig 12: Final Output

## VI. CONCLUSION

In this work, we answered three research questions focused on Onion Service traffic classification. We evaluated the applicability of supervised machine learning models to classify Onion Service traffic from other Tor traffic. We extracted fifty features from each traffic trace and used that feature set as input to the machine learning classifiers. Our results showed that KNN, RF, and SVM classifiers have the ability to distinguish Onion Service traffic from Tor traffic with a 99% accuracy. Then, we tried to identify whether state-of-the-art Website Fingerprinting defences affect the classifiability of Tor traffic. These defences introduce different modifications to try and obfuscate information leakage from traffic, and we evaluated how those changes affect the Onion Service traffic classification. Our experiments showed that the above classifiers, combined with our feature set, reduce the performance for Onion Service traffic classification. However, we observed that the modified Tor traffic is still distinguishable. Moreover, we used three feature selection metrics, namely, information gain, Pearson's correlation, and Fisher Score, to identify the top features for this task. Those top features were able to provide >98% success for classifying Onion Service traffic from Tor traffic. However, they could not provide such good results when modified Tor traffic traces were used.

## VII. FUTURE SCOPE

Further research needed to assess classifier performance when Tor traffic is modified with tools like WTFPAD or TrafficSliver, potentially affecting its differentiation from non-Tor traffic. The project can investigate how various features affect classifier performance in identifying Onion Service traffic, offering insights into their impact on classification effectiveness. Additional research is needed to comprehend the consequences of easily identifying Tor and Onion Service traffic, particularly regarding traffic monitoring and potential restrictions imposed by governments and sensitive institutions. The project can also explore the development of more advanced techniques to improve the accuracy of identifying Onion Service traffic, even in the presence of obfuscation techniques.

## REFERENCES

- [1] R. Dingleline, N. Mathewson, and P. Syverson, "Tor: The Second Generation Onion Router," in Proceedings of the 13th USENIX Security Symposium, SSM'04, (San Diego, CA, USA), pp. 303–320, August 2004.
- [2] M. AlSabah, K. Bauer, and I. Goldberg, "Enhancing Tor's Performance Using Real-Time Traffic Classification," in Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, (New York, NY, USA), pp. 73–84, October 2012.
- [3] A. H. Lashkari., G. D. Gil., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of Tor Traffic using Time based Features," in Proceedings of the 3rd International Conference on Information Systems Security and Privacy, ICISPP 2017, (Porto, Portugal), pp. 253–262, February 2017.
- [4] M. Kim and A. Anpalagan, "Tor Traffic Classification from Raw Packet Header using Convolutional Neural Network," in 1st IEEE International Conference on Knowledge Innovation and Invention, ICKII 2018, (Jeju Island, South Korea), pp. 187–190, July 2018.
- [5] G. He, M. Yang, J. Luo, and X. Gu, "Inferring Application Type Information from Tor Encrypted Traffic," in Proceedings of the 2014 Second International Conference on Advanced Cloud and Big Data, CBD '14, (NWWashington, DC, USA), pp. 220–227, November 2014.
- [6] A. Montieri, D. Ciunzo, G. Aceto, and A. Pescapé, "Anonymity Services Tor, I2P, JonDonym: Classifying in the Dark (Web)," IEEE Transactions on Dependable and Secure Computing, vol. 17, pp. 662–675, May 2020.
- [7] "WCry Ransomware Analysis." <https://www.secureworks.com/research/wcry-ransomware-analysis>, May 2017. Accessed: 2023-04-26 [Online].
- [8] "Keeping a Hidden Identity: Mirai C&Cs in Tor Network." <https://blog.trendmicro.com/trendlabs-security-intelligence/keeping-a-hidden-identity-mirai-ccs-in-tor-network/>, July 2019. Accessed: 2023-04-26 [Online].



- [9] "Global action against dark markets on Tor network." <https://www.europol.europa.eu/newsroom/news/global-action-against-dark-markets-tor-network> , November 2014. [Online].
- [10] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an Efficient Website Fingerprinting Defense," in Proceedings of the 21st European Symposium on Research in Computer Security, ESORICS2016, (Heraklion, Greece), pp. 27–46, September 2016.
- [11] T. Wang and I. Goldberg, "Walkie-Talkie: An Efficient Defense against Passive Website Fingerprinting Attacks," in Proceedings of the 26th USENIX Security Symposium, SEC'17, (Vancouver, BC, Canada), pp. 1375–1390, August 2017.
- [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 Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS'20, (New York, NY, USA), pp. 1971–1985, November 2020.
- [13] J. Hayes and G. Danezis, "K-Fingerprinting: A Robust Scalable Website Fingerprinting Technique," in Proceedings of the 25th USENIX Conference on Security Symposium, SEC'16, (Austin, TX, USA), pp. 1187–1203, August 2016.
- [14] X. Bai, Y. Zhang, and X. Niu, "Traffic identification of tor and web-mix," in Proceedings of the 2008 Eighth International Conference on Intelligent Systems Design and Applications- Volume 01, ISDA '08, (Kaohsiung, Taiwan), pp. 548–551, November 2008.
- [15] O. Berthold, H. Federrath, and S. Köpsell, "Web MIXes: A System for Anonymous and Unobservable Internet Access," in Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability (H. Federrath, ed.), vol. 2009 of Lecture Notes in Computer Science, (Berkeley, CA, USA), pp. 115–129, July 2000.



10.22214/IJRASET



45.98



IMPACT FACTOR:  
7.129



IMPACT FACTOR:  
7.429



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24\*7 Support on Whatsapp)