SN Computer Science (2022) 3:126
https://doi.org/10.1007/s42979-022-01017-z

ORIGINAL RESEARCH

Identification and Counting of Hosts Behind NAT Using Machine Learning

Sanjeev Shukla[1] · Himanshu Gupta[1]

Received: 29 September 2021 / Accepted: 1 January 2022 / Published online: 11 January 2022

Abstract
NAT is a process of deploying a large number of hosts (typically in a LAN) to use a single or group of limited public IP address. The anonymity that NAT provides is a privilege for legitimate user, but it is a convenient place for malicious user/attackers to hide without being detected which helps them to perform Denial of Service (DoS) attack. Thus, identifying host hidden behind NAT propagating malicious behavior is necessary for forensic investigation. In this paper, we propose a Machine Learning (ML)-based algorithm to identify and count total number of host behind NAT. This is achieved by identifying patterns in network traffic through flow level statistics and detecting anonymity of hosts. The novelty of proposed work lies in counting host and comparing the results with state-of-art methods of conventional signature-based approach. The results obtained show higher accuracy (93% average using three datasets) which significantly outperforms signature-based approach results (75% approx.). Furthermore, the model is also tested by feeding it with malicious network traffic dataset to identify and count whether the traffic is originating from single or multiple hosts. The higher accuracy (92.2% approx.) obtained reflect that the model can predict and count correct number of malicious host with least false positives. In this work, we identify and count number of hosts behind NAT by applying ML-based techniques on statistical net- flow records. The advantage of the proposed approach is that privacy-related issue while capturing the header does not arise and it has no dependency on OS, IP address, or port numbers.

Keywords NAT · DoS · ML · Network security

Introduction

Network Address Translation (NAT) is a very common device deployed by Internet Service Providers (ISPs), to provide Internet services. NAT [1] is the process of replacing private IP address of multiple host with one public IP address. This is done to deploy a lot of systems in a small private Local Area Network (LAN) behind NAT gateway to use the Internet through the NAT gateways public IP interface. The machines behind gateway use private IP addresses for intranet activities and use gateways public address for Internet. NAT is thus a vital element in any network setup and has gained enormous popularity for providing solution to shortage of IP addresses and privacy issues by providing users in the private LAN anonymity. Installed at the entry and exit point of private and public network, it also assists in some security issues by hiding inner network topology, network monitoring, user anonymity, and content filtering.

This article is part of the topical collection "Cyber Security and Privacy in Communication Networks" guest edited by Rajiv Misra, R K Shyamsunder, Alexiei Dingli, Natalie Denk, Omer Rana, Alexander Pfeiffer, Ashok Patel, and Nishtha Kesswani.

✉ Sanjeev Shukla
sanjufcc@iitr.ac.in

Himanshu Gupta
himanshugpt3@gmail.com

1 Department of Computer Science, Indian Institute of Technology, Roorkee, Uttarakhand 247667, India

Motivation

A white paper published by a security service provider company illustrates in its study "The Myth of NAT as Security provider" that large number of attacks and threats emanate from ISP network using NAT. This is also found in mobile networks [2]. Similarly, recent research article published in ACM on Sep 2021 discusses vulnerability of NAT in DOS attacks [3]. Therefore, motivation of this work is to develop an ML model which identifies the hosts behind NAT device and helps in counting them as well.

Challenges in NAT

NAT thus assists in sorting some of the challenges in terms of security and privacy issues, yet has its own inherent security flaws. The biggest problem is identifying individual hosts hidden behind the public interface of NAT device [4]. For malicious uses, it provides complete anonymity. This anonymity that NAT provides is a privilege for legitimate users, but it is also safe haven for cyber-attacks or malicious users.

Need for Host Identification

The challenge to identify or trace back the exact host in private LAN (which is the source of cyber-attack or malware propagation) from outside the network is difficult. Every attempt to locate the source of attack from Wide Area Network (WAN) can lead them to the public interface of the network gateway (NAT) which acts as the bottleneck in detecting source. To further detect and find the exact host without the help of Network Admin are not possible. Also, nowadays, ISP, while providing Internet connectivity to users, offers services based on the number of active connections (apart from other bandwidth-based plans). In case of the number of active user-based connections (e.g., ten active uses max), ISP needs to know the number of active users behind NAT to ensure that they do not cross the paid user connection quota. In such scenarios, counting of host is essential. Other applications like streaming server also require host count to serve its users better, as the server offer limited user connection due to server overload. In case of NAT user, a single user request (hiding large number of anonymous users) may overwhelm the server with Real-Time Streaming Protocol (RTSP) request denying other users server access. This may mimic a DOS attack, and user counts are essential for such services [5, 6].

Paper Contribution

The traditional ML-based approach applied till now is to detect rouge NAT or NAT device from network traffic. Counting of host has not been done so far. Our paper proposes an ML-based algorithm to identify and count total number of host behind NAT. This is done by looking for patterns in network traffic through flow level statistics and detecting anonymity of hosts. The novelty of the proposed work lies in counting the total number of hosts behind NAT and comparison of its results with the state-of-the-art method of signature-based approach. Our key contributions in this paper are:

1. Identifying and counting total number of hosts hidden behind NAT using proposed ML-based approach.
2. Feeding the model with malicious host dataset to count number of malicious hosts. This will help in cases where malicious traffic is detected, but we do not know whether it is coming from single host or multiple host.
3. Comparative analysis with other state-of-art techniques to establish the efficacy of the model.

Thus, the objective of current work is to develop an ML model which readily detects hosts behind NAT device (in private LAN) and helps in counting total number of hosts. The significance of such model is to accurately predict the anonymity of hosts based on patterns from network traffic using flow level statistics.

Paper Organization

The paper is structured as follows. The section "Related Works" discusses past work done in the field of host detection and identification in NAT environment and compares with our proposed work. The section "The Proposed Method" dwells in detail the proposed work, flow of work, explanation, and data set used. The section "Methodology Applied" shows the methodology applied, classifier selection, performance metrics, etc. The section "Results and Discussion" discusses the results obtained, and finally, the section "Conclusion and Future Work" contains conclusion and provides some future work directions.

Related Works

Host identification techniques—host detection—can be achieved by mainly two approaches, active probing or passive probing. Active probing technique sends a set of queries to the targeted systems and analyzes their response. It requires two-way communication with the targeted host and may not be allowed in all network configurations. Also, it is intrusive, detectable, and consumes network resources. On the other hand, passive probing technique captures the network traffic to be analyzed later. It does not generate its own traffic, and passively monitors and observes network traffic. Therefore, it is non-intrusive, undetectable and retrospectively analyzes the captured network traffic. Hence, we will be only focusing on passive techniques. Passive techniques are further classified as signature-based approach and ML-based approach.

Signature-Based Techniques

Signature-based technique for detection of hosts behind NAT is to find a set of parameters (signature) that can define the host uniquely in the absence of IP address. The whole process is to capture the network traffic passively and perform header analysis to identify host. In the past, many approaches have been suggested by researchers to detect host behind NAT like Cohen [7] presented a model that used the correlation between different protocol fields to identify the source machine, Maier et al. [8] used IP TTL and HTTP user-agent strings to estimate the number of hosts connected to an NAT device, Mongkolluksamee et al. [9] proposed a passive method for counting hosts having similar IP address, i.e., NATed device, Wicherski et al. [10] proposed a lightweight real-time flexible framework NATFILTERED which uses times-tamps (TCP) to identify hosts without having dependency on IP addresses, Paul et al. [11] suggest a TCP SYN-based approach for identification of malicious activity by unauthorized OS used in virtual machines(VM) environments, and Park et al. [12] suggested a technique to detect total hosts behind NAT along with their OS type using header fields of TCP/IP protocols.

ML-Based Technique

Table 1 shows in detail literature survey done on ML-based approach to detect NAT and host behind it. Abt et al. [13] proposed to detect rouge NAT device using behavior statistics extracted from netflows. The binary classifier uses nine distinct features to classify the flows as NAT or Non-NAT. Verde et al. [14] proposed a fingerprinting framework to identify host by taking a sample of hosts and profiling them to train HMM classifier. The encoded netflow into training set is used by user detector module to aggregate time interval and classify users as connected or not. Limitation of such approach is that first users have to be trained, and this reduces its applicability and also its performance degrades on using TOR or VPN tunneling.

Gokcen et al. [15] proposed ML-based approach to detect malicious behaviors of hosts behind NAT devices from network traffic flows. It deploys C4.5 algorithms which is a decision tree classifier and also uses Naive Bayes statistical classifier which applies Bayes theorem to get conditional probability of a given class. It further uses WEKA, an open-source tool and measures the performance by confusion matrix parameters. It works well without application (payload) information and in encrypted environment as well. Komark et al. [16] proposed to passively utilized IP features present in HTTP access logs to detect NAT behaviors. They extracted all the unique features from HTTP logs like

Table 1 Past work in host and NAT detection using ML

References	Objective	Classifier applied	NAT detection	Host profiling	Host counting	Dataset type	No of features used	Accuracy (Avg %)
Abt et al. [13]	To detect a rouge NAT device using behavior statistics	SVM and DT	Yes	No	No	Netflows	9	89.95
Varde et al. [14]	Fingerprinting users Behind NAT	HMM	No	Yes	No	Netflows	–	90
Gokcen et al. [15]	Identify NAT behavior by analyzing traffic flows	NB and C 4.5	Yes	No	No	Netflows	40	DR = 98
Komarket al. [16]	NAT Detection using Statistical behavior analysis	SVM	Yes	No	No	HTTP access logs	8	95
Khatouni et al. [17]	NAT detection and host identification	DT	Yes	No	No	Netflows	10	F1 score = 96
Lee et al. [18]	NAT device identification using port response pattern	DT	Yes	No	No	Netflows	–	F1 score = 90
Proposed method	Counting total number of hosts behind NAT	DT	No	No	Yes	Netflows	14	F1 score = 92

user-agents, OS and its version, contacted IP Addresses, number of persistent connections, browser family and its versions, number of HTTP request sent, number of upload and download bytes, etc., for analysis and NAT identification.

Khatouni et al. [17] presented an ML-based model for passively measuring NAT behavior using Netflow statistics. They compared ten Machine learning algorithms on eight different Netflow features, four features from literature [15], and four from popular netflow softwares to conclude that Tranalyzer feature set and its performance were much better than other software. Their results concluded that the solution provided by Decision Tree (DT) classifier using Tranalyzer achieved highest F1 score with minimal computational cost.

Lee et al. [18] proposed to identify NAT device using port response pattern. This is achieved by applying supervised learning on model which utilizes asymmetric pattern of port responses by NAT and Non-NAT device. Limitation of this approach is large time requirement in collecting port responses, active probing challenges like blocking of probes by firewall or IDS considering them as threats.

The Proposed Method

In this work, our aim is to identify the number of hosts hidden behind NAT device and count them. We then perform comparative analysis of traditional signature-based method and ML-based method. We wanted to compare our work with similar ML-based methods as well, but most of the papers (as shown in Table 1) detect NAT and do not count host. For the proposed method, we have used Decision Tree (DT) classifier after comparing its performance with eight multi-class classifier. For signature-based method, we have written python code based on literature [9, 10, 12] as state-of-art technique. For both the approaches, we have deployed the same dataset.

Work Flow Process

The block diagram representing the basic workflow process is shown in Fig 1 and steps involved in the section "The Proposed Method" are as follows:

1. The first step is to capture network traffic of the sample network (or test setup). This is done using wireshark and the file thus obtained is in pcap format.
2. The pcap file is fed in tranalyzer application to extract statistical features. Tranalyzer is given 105 features by default.
3. Next, we apply pre-processing and feature selection method to extract most significant features of the dataset.

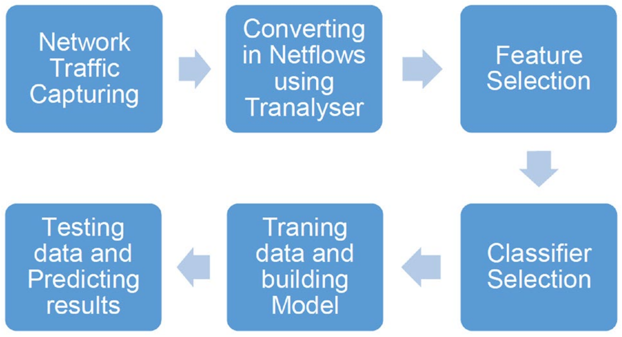

Network Traffic Capturing
Converting in Netflows using Tranalyser
Feature Selection
Classifier Selection
Traning data and building Model
Testing data and Predicting results

Fig. 1 Proposed work flow block diagram

4. Different multi-class classifiers are tested to find the most suitable classifier for our problem statement.
5. After selection of classifier, the dataset is trained to build the model for testing.
6. Finally, test dataset is applied to obtain the results from the model.

Data Sets

In the proposed work, we have taken three different datasets. Two are from online sources and one from our lab setup. First data set is available online and belongs to ICS Lab [19]. It contains non-NAT-ted traffic flows from ICS Lab with 1116 different hosts. We named it as ICS data set. The dataset contains a pcap file with size of around 400 MB. Tranalyser was used to convert the pcap file to csv and key distinct features were extracted. This dataset after pre-processing contains 4.6 lakh flows which is a large dataset; hence, we choose this for training our model.

Second online resource is a pcap file available at malware-traffic-analysis (MTA) website [20]. This is a small dataset (referred as MTA Dataset) which contains NAT-ted and non-NAT-ted traffic flow from 143 malicious hosts. The objective of testing this dataset is to see if the proposed method can detect and count malicious host or not.

Third data set was generated in our lab with 73 different hosts for a period of 6 h. Our setup is shown in Fig 2 which consists of a switch and 73 PCs each with 8GB RAM and an i7 hexa core processor (clock speed 4.3 GHz). These 73 PCs were used for generating network traffic and capturing devices were places before and after NAT device to collect both NAT-ted and non-NAT-ted traffic flow. Traffic was generated using a script which opens multiple URLs in web browser. The data were collected in pcap format using wireshark and key parameters were extracted using tranalyzer. The extracted parameters were pre-processed and converted to csv file. We named this data set as LAB data set.

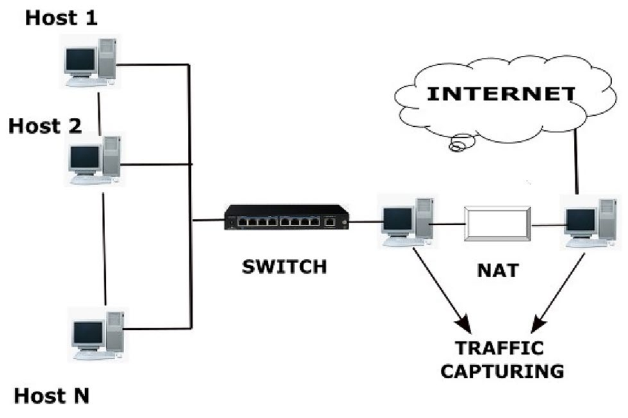


Fig. 2 Setup of our lab

Softwares Used

- Tranalyzer: It is a flow generator tool which supports processing of large packet dumps. It provides functionality to generate statistics from pcap files. There are numerous softwares which are present in market including Netmate, Argus, Tstat, etc. We used Tranalyzer in our work, because according to literature [17], it gives better results as compared to other softwares.
- Wireshark: It is free as well as open-source packet analyzer which can be used for traffic capturing and analyzing. We used wireshark as a tool for capturing the data packets in pcap format for our LAB dataset.
- Weka: Weka is an open-source tool which contains a collection of visualization tools and algorithm for data analysis and predictive modeling. It also supports multiple tasks like clustering, classification, and regression.

Feature Selection

The dataset thus obtained is converted into netflows. This is done using Tranalyzer which converts the pcap file into distinct statistical feature based flows. The advantage of using flow-based method is that we do not require IP addresses and port numbers any more. Hence, any type of bias that NAT implementation brings is removed in proposed method.

Features Used in Proposed ML-Based Approach

It is a significant step in host identification process. Application of Tranalyzer on dataset in pcap format yields 105 statistical features.

Pre-processing Pre-processing includes data preparation, filtering, cleaning, feature extraction, and selection process. Data set which is not carefully examined can produce misleading results. Thus, we pre-processed our data sets by removing the redundant features and constant values. Also, features present in string format were converted to the corresponding integer value to maintain the importance of each feature. It therefore resulted in subset of 47 features.

Feature Selection Feature selection is the most critical part for any data model. Here, our aim is to select the most optimal features among 47 features, which should be independent of user behavior and network environment to generate a high-efficiency data model which should be robust enough to work in different environments. To do this, we have used the combination of two methods: (1) Filtering Method and (2) Wrapper Method. The first one is the filtering method which was required prior to the wrapper method to make the wrapper method computationally efficient. Under Filtering method, we tried Chi-Square (CS) and Information gain (IG) techniques and we found that CS results were not appropriate for our problem statement due to its dependent on significance level. Therefore, we choose IG which was more appropriate in terms of results for our problem. Also, as per literature, most of the authors have shown IG as a good technique for selecting the optimal attributes with reference to our problem statement [17]. Furthermore, a brief comparison was also done to understand the discrepancies between Gini Index (GI) and IG based on Entropy quantifier. While GI facilitates the bigger distributions and is easy to implement, whereas the IG favors lesser distributions having small count with multiple specific values. GI is predominantly used in CART algorithms, while IG is used in ID3, C4.5, and J48 algorithms. Also, GI operates on the categorical target variables in terms of “success” or “failure” and performs only binary split; IG, on the other hand, computes the difference between entropy before and after the split and indicates the impurity in classes of elements. IG is thus applied to calculate the normalized average impurity using Eq. (1)

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{values }(A)} \frac{Sv}{S} \text{ Entropy } Sv, \quad (1)$$

where Values(A) is the all possible values of attribute A, and Sv is the subset of S for which attribute A has value v. Entropy is calculated using Eq. (2)

$$\text{Entropy}(S) = -\left(P \oplus \log_2 P \oplus + P \ominus \log_2 P \ominus\right), \quad (2)$$

where P is the portion of positive examples and P is the portion of negative examples in S.

Figure 3 shows the ranking of all the features based on their information gain values where the x-axis shows the feature number (each feature out of 47 is denoted by a number) and the y-axis shows the Info Gain value. As we can observe from Fig 3, the gain value dips sharply after 22 features. Thus, we select top 22 features which have the highest impact on our model.

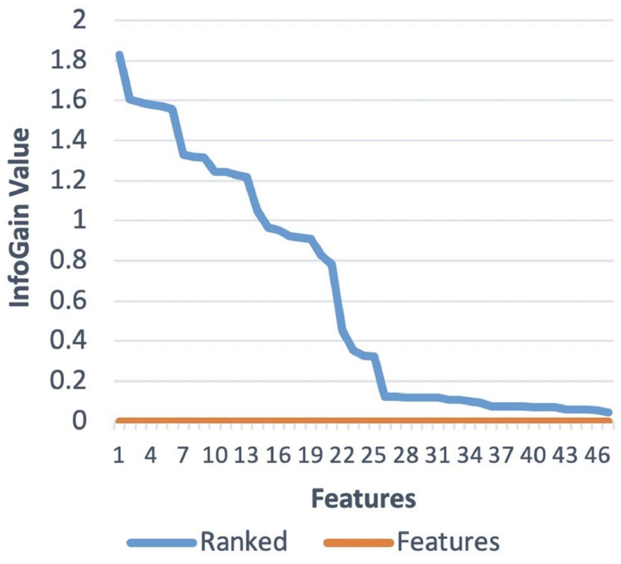


Fig. 3 Feature ranking

After the filtering method, we move toward wrapper method. In wrapper method, we focused on genetic algorithm with one-point crossover which aims in finding the most optimal set of features for our problem. The block diagram representing the wrapper method process is shown in Fig 4 and steps involved in this method are as follows:

1. In the first step, we get initial set of features through filter method.
2. In the second step, a portion of features are selected from all the available attributes left after filter method using crossover technique.
3. In the third step, ML algorithm is implemented on the subset of features selected through crossover.
4. In the fourth step, performance of the algorithm using selected features is analyzed and the process is repeated from 2nd step until we get optimal set of features.

We found that wrapper method resulted in a subset of 14 features, as shown in Table 2, which gives the best result for our problem statement. Therefore, we selected these 14 features for the given problem as it has improved the overall performance of the model as compared to 47 features.

Table 2 Features selected

Feature	Description
dstPort	Destination port
tcprttAckTripMax	TCP ACK trip maximum
tcpRTTAckTripAve	TCP ACK trip average
tcpAveWinSz	TCP average effective window size
tcpMaxWinSz	TCP maximum effective window size
tcpInitWinSz	TCP initial effective window size
tcpMinWinSz	TCP minimum effective window size
tcpRTTAckJitAve	TCP ACK round trip average jitter
tcpAckFaultCnt	TCP ACK number fault count
tcpWinSzThRt	TCP count ratio below WINMIN threshold
flowInd	Flow index
dstPortClassN	Port based classification of destination port
pktAsm	Packet stream asymmetry
tcpWS	TCP window scale factor

Methodology Applied

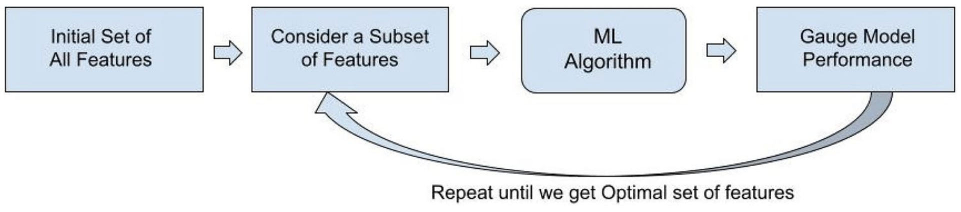
For Proposed Approach

In this, we use ICS data set to train our model which contains traffic flow from 1116 hosts. We identify each host with the help of source IP address and mark them as a unique class. Once classes are assigned to each flow, we train our model with the help of ICS data set. After training our data model, we have used different data sets which contains NATted traffic flow from multiple hosts to test our trained model. Our trained model then classifies each flow to a class with the help of all the selected features. Unique classes are then counted to count number of hosts. To start with, we first select an appropriate classifier.

Classifier Selection

The choice and selection of ML classifier for classification and prediction of data are based on its performance. A multi-class classifier is selected for prediction as there are multiple classes. Next, eight multi-class ML classifiers are compared on ICS dataset with 14 features set (as obtained from feature extraction) where k-fold cross validation ($k = 10$) is applied. Table 3 presents the classifier used for comparison.

Fig. 4 Wrapper method flow diagram



Initial Set of All Features
Consider a Subset of Features
ML Algorithm
Gauge Model Performance
Repeat until we get Optimal set of features

Table 3 List of multi-class classifiers

S. no.	Algorithm	Short form
1	Naive Bayes	NB
2	Hoeffding Tree	HT
3	Random Tree	RT
4	Decision Tree	DT
5	Random Forest	RF
6	K Nearest Neighbor	KNN
7	Multilayer Perceptron	MLP
8	Support Vector Machine	SVM

All the chosen classifier performance is evaluated with precision, recall, and F1-score values, as shown in Fig 5, respectively. The comparison thus shows Decision Tree (DT) and Random Forest (RF) classifiers outclassing the remaining classifier. The results of both are quite similar, but the model building time (training time) is vastly different. It took 18 s for DT to build model. RF model built-up time was 182 s and improvement in performance is just 0.1% (compared to DT performance). The system configuration of the machine used for testing our ML model is: OS—Windows 10, Processor—Intel(R) Core(TM) i5-7200U CPU @ 2.7 GHz, system—64-bit OS and processor, RAM—16 GB and HDD—1 TB. Hence, DT is chosen, as the model build-up time is very less compared with RF and the results obtained are faster with nearly similar performance as RF.

Training and Testing Model

Once the classifier is selected, we use the ICS dataset to train the model. First, we apply tenfold cross validation. The model is further tested with MTA and LAB dataset to check its performance.

Performance Metrics Applied

The performance is evaluated using four metrics, which are calculated based on confusion matrix. True Positive (TP) is result in which model has correctly predicted netflows to its host. False Positive (FP) is a result in which model incorrectly predicts netflow of other hosts to its host. False Negative (FN) is a result in which model incorrectly predicted its netflow to some other host. True Negative (TN) is considered as 0, because there is no netflow which is rejected. The details about metrics that have been considered for performance evaluation are as follows:

- Precision: It defines as the ratio of TP to the total sum of TP and FP

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{3}$$

- Recall: It is defined as the capability of a classifier to detect all available positive samples. It is formulated as

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{4}$$

- F1 Score: It is defined as HM (Harmonic Mean) of Recall and Precision which is given by

$$f1\ \text{Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{5}$$

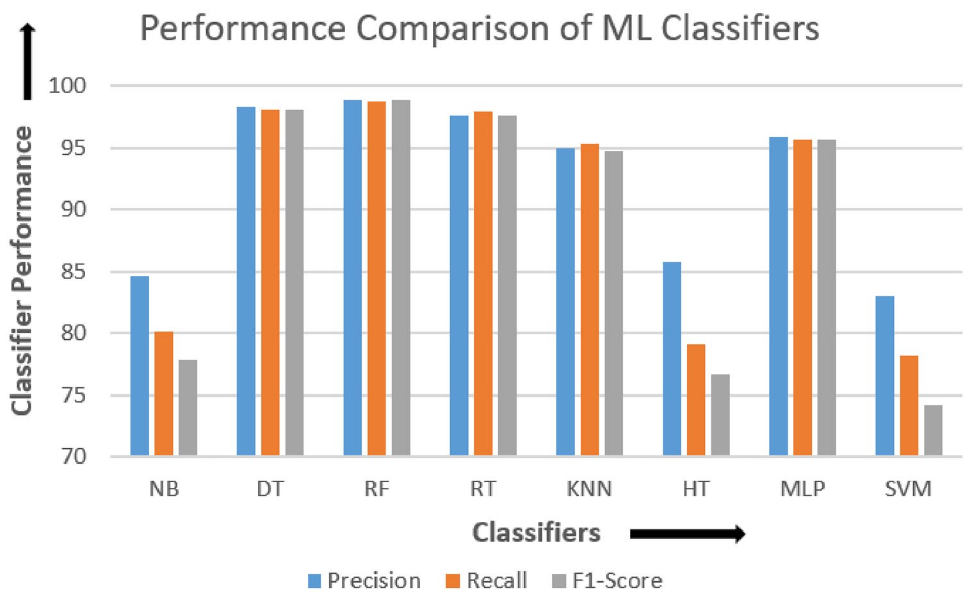


Fig. 5 Comparison of different machine learning classifiers

Table 4 Results of proposed approach

Dataset	Precision (%)	Recall (%)	F1 score (%)	Accuracy (%)
MTA data set	92.1	92.6	92.1	92.2
LAB data set	87.0	89.0	88.0	89.0

- Accuracy: It is the ratio of sum of true positives and true negatives to the total number of packets

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{6}$$

Results and Discussion

In this section, we present the result obtained by proposed method and compare it with signature-based method.

Performance Evaluation of Proposed ML Approach

The performance of proposed method is evaluated using precision, recall, F1 score, and accuracy, as shown in Table 4. The results thus obtained show that the datasets used have performed better with average accuracy of 91%.

When ICS dataset is used for both training and testing, we get an average accuracy of 98% which shows when training and testing are done with the same dataset, and it performs much better. Performance of LAB dataset on the other hand shows slight reduction, as shown in Table 4. This reflects, because the training dataset (ICS dataset) is different form the testing dataset (LAB dataset). Thus, if training and testing datasets are different, the performance of model reduces. Furthermore, the MTA dataset having malicious traffic is identified correctly to predict correct number of hosts. This means if malicious traffic is fed to our algorithm, it will determine whether malicious traffic is coming from single host or multiple host (by counting total number of hosts).

Comparative Analysis of Proposed Method with Signature-Based Approach

Performance evaluation of state-of-art signature-based approach (using the same dataset) when implemented through our code resulted in accuracy of 75% (max) which is similar to literature results (approximately). A comparative performance evaluation using accuracy of both the methods clearly shows in Fig 6 that the proposed method (in blue) based on ML outperforms the signature-based approach.

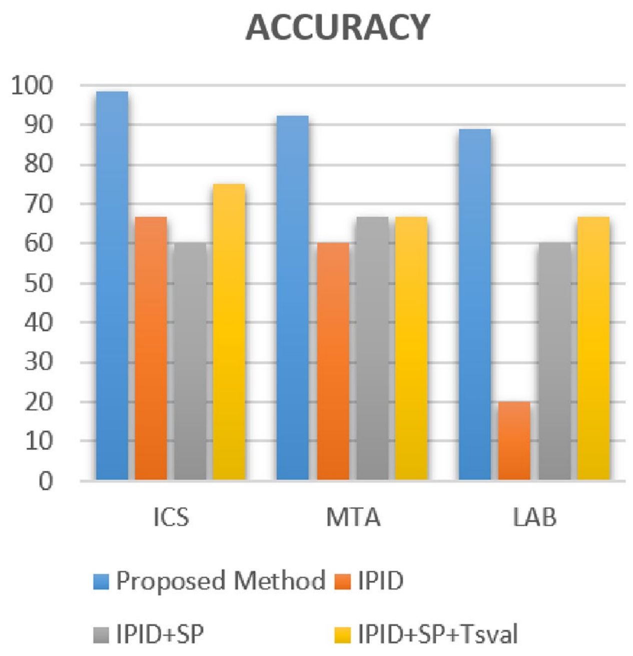


Fig. 6 Comparison of signature and proposed method

Comparison Points in the Benchmarks and Proposed Framework

As shown in Table 5, comparison results show that most of the benchmark studies obtained the score of 0% or 33.33% whereas our proposed frame work obtained the score of 66.66%. The study of previous benchmarks was mainly focused on the detection of NAT device, whereas our proposed framework focuses on the hosts present behind the NAT device. Also, our proposed framework is robust enough to handle the malicious traffic which previous benchmark studies lacked.

Conclusion and Future Work

In this paper, we describe an ML-based technique which is OS independent and counts the number of hosts present behind NAT by analyzing statistical network traffic flows. Three different dataset is taken to conduct test. Statistical features were obtained by processing the dataset through Tranalyzer. Applying pre-processing and feature selection method results in obtaining 14 features of most significance. Different multi-class classifiers were compared (based on their performance) and Decision Tress was found the best among them. The results thus obtained show better results with average performance of 91% for all the datasets used. The model also shows good results in counting malicious host when fed with malicious traffic. This will help in determining whether malicious traffic is coming from single host

Table 5 Benchmarking checklist

Checklist issues	Abt et al. [13]	Varde et al. [14]	Y.Gokeen et al. [15]	Komark et al. [16]	Khatouni et al. [17]	Lee et al. [18]	Proposed work
Handling NAT device detection	✓	x	✓	✓	✓	✓	x
Handling Host count	x	x	x	x	x	x	✓
Handling malicious traffic	x	x	x	x	x	x	✓
Score	33.33%	0%	33.33%	33.33%	33.33%	33.33%	66.66%
Difference	33.33%	66.66%	33.33%	33.33%	33.33%	33.33%	–

or multiple host. Also, the results outperform when compared with signature-based technique. The limitation of current technique is that if n classes are used for training the model, it will predict m classes where $m \leq n$. This limitation can be overcome by increasing n (by training maximum host in a network).

Future work will include detecting malicious traffic and counting total number of benign and malicious hosts hidden behind NAT from the network traffic. The current work can also be made online by uploading it on cloud for periodic training and updating the model for a dynamic network environment.

Funding Not applicable.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Ishikawa Y, Yamai N, Okayama K, Nakamura M. An identification method of pcs behind nat router with proxy authentication on http communication. In: 2011 IEEE/IPSJ international symposium on applications and the internet, IEEE; 2011. pp. 445–450.
2. The Myth of Network Address Translation as Security. 2016. https://www.f5.com/services/resources/white-papers/the-myth-of-network-address-translation-as-security. Accessed 10 Feb 2016.
3. Akashi S, Tong Y. A vulnerability of dynamic network address translation to denial-of-service attacks. In: 4th International conference on data science and information technology, ACM; 2021. pp. 226–230.
4. Tang F, Kawamoto Y, Kato N, Yano K, Suzuki Y. Probe delay based adaptive port scanning for Tot devices with private IP address behind NAT. J IEEE Netw. 2020;34(2):195–201.
5. Tekeoglu A, Altiparmak N, Tosun A. Approximating the number of active nodes behind a nat device. In: 2011 Proceedings of 20th international conference on computer comm networks (ICCCN), IEEE; 2011. pp. 1–7.
6. Meidan Y, Sachidananda V, Peng H, Sagron R, Elovici Y, Shabtai A. A novel approach for detecting vulnerable Iot devices connected behind a home NAT. J Comput Secur. 2020;97: 101968.
7. Cohen MI. Source attribution for network address translated forensic captures. J Digit Investig. 2009;5:138–45.
8. Maier G, Schneider F, Feldmann A. Nat usage in residential broadband networks. In: International conference on passive and active network measurement, Springer; 2011. pp. 32–41.
9. Mongkolluksamee S, Fukuda K, Pong P. Counting natted hosts by observing TCP/IP field behaviors. In: 2012 IEEE international conference on communications (ICC), IEEE; 2012. pp. 1265–1270.
10. Wicherski G, Weingarten F, Meyer U. IP agnostic real-time traffic filtering and host identification using TCP timestamps. In: 38th IEEE conference on local computer networks, IEEE; 2013. pp. 647–654.
11. Tyagi R, Paul T, Manoj B, Thanudas B. Packet inspection for unauthorized OS detection in enterprises. IEEE Secur Priv. 2015;13(4):60–5.
12. Park H, Shin RBS, Lee C. Identification of hosts behind a NAT device utilizing multiple fields of IP and TCP. In: 2016 International conference on information and communication technology convergence (ICTC), IEEE; 2016. pp. 484– 486.
13. Abt S, Dietz C, Baier H, Petrovic S. Passive remote source NAT detection using behavior statistics derived from netflow. In: IFIP international conference on autonomous infrastructure, management and security, Springer; 2013. pp. 148–159.
14. Verde N, Ateniese G, Gabrielli E, Mancini L, Spognardi A. No nat'd user left behind: fingerprinting users behind NAT from netflow records alone. In: 2014 IEEE 34th international conference on distributed computing systems, IEEE; 2014. pp. 218–227.
15. Gokcen Y, Foroushani VA, Heywood A. Can we identify NAT behavior by analyzing traffic flows? In: 2014 IEEE security and privacy workshops, IEEE; 2014. pp. 132–139.
16. Komarek T, Grill M, Pevny T. Passive NAT detection using http access logs. In: 2016 IEEE international workshop on information forensics and security (WIFS), IEEE; 2016. pp. 1–6.
17. Khatouni A, Zhang L, Aziz K, Zincir I, Heywood N. Exploring NAT detection and host identification using machine learning. In: 2019 15th International conference on network and service management (CNSM), IEEE; 2019. pp. 1–8.
18. Lee S, Kim SJ, Lee J, Roh B. Supervised learning-based fast, stealthy, and active NAT device identification using port response patterns. Published in Symmetry Journal of MDPI. 2020;12(9):1444. https://doi.org/10.3390/sym12091444.
19. Data Set of Industrial Cyber Security Conference, ICS LAB. (2015). https://4sics.se/. Accessed 1 Feb 2020.
20. Data Set from the Malware Analysis Website. 2017. https://malware-traffic-analysis.net/. Accessed 9 Feb 2020.