9-2-2020

Supervised Learning-Based Fast, Stealthy, and Active NAT Device Identification Using Port Response Patterns

Seungwoon Lee
Ajou University, swleeyg@ajou.ac.kr

Si Jung Kim
University of Nevada, Las Vegas, si.kim@unlv.edu

Jungtae Lee
Ajou University, jungtae@ajou.ac.kr

Byeong-hee Roh
Ajou University

symmetry

MDPI

Article

Supervised Learning-Based Fast, Stealthy, and Active NAT Device Identification Using Port Response Patterns

Seungwoon Lee [1], Si Jung Kim [2], Jungtae Lee [1] and Byeong-hee Roh [1,*]

[1] Department of Computer Engineering, Ajou University, Suwon 16499, Korea; swleeyg@ajou.ac.kr (S.L.); jungtae@ajou.ac.kr (J.L.)
[2] Howard R. Hughes College of Engineering, University of Nevada, Las Vegas, NV 89154, USA; sj.kim@unlv.edu
* Correspondence: bhroh@ajou.ac.kr; Tel.: +82-31-219-1601

Received: 18 July 2020; Accepted: 24 August 2020; Published: 2 September 2020

Abstract: Although network address translation (NAT) provides various advantages, it may cause potential threats to network operations. For network administrators to operate networks effectively and securely, it may be necessary to verify whether an assigned IP address is using NAT or not. In this paper, we propose a supervised learning-based active NAT device (NATD) identification using port response patterns. The proposed model utilizes the asymmetric port response patterns between NATD and non-NATD. In addition, to reduce the time and to solve the security issue that supervised learning approaches exhibit, we propose a fast and stealthy NATD identification method. The proposed method can perform the identification remotely, unlike conventional methods that should operate in the same network as the targets. The experimental results demonstrate that the proposed method is effective, exhibiting a F1 score of over 90%. With the efficient features of the proposed methods, we recommend some practical use cases that can contribute to managing networks securely and effectively.

Keywords: Network Address Translation (NAT); supervised learning; port response pattern; decision tree; network administration

1. Introduction

As the Internet grows dramatically, the allocation of a unique Internet Protocol version 4 (IPv4) address to each device connected to the Internet has become a problem. Although IPv6 has been introduced to solve the IP address exhaustion problem, the complete adoption of IPv6 has been delayed, in part owing to network address translation (NAT) [1]. NAT enables multiple hosts in a private network to access the Internet with one public IP address. With NAT, IPv4 is likely to dominate IPv6 for a long time.

Apart from these advantages, NAT exhibits the following problems. Numerous hosts with private IP addresses can connect to the Internet through a NAT device (NATD). However, it is challenging for an entity outside the NATD to identify how many hosts are behind the NATD (called NATHs) [2]. A large number of NATHs can generate tremendous traffic that can overload the network. Moreover, a few of these can conduct malicious behaviors that can impair the Internet. Because the network administrator acknowledges this traffic to be originating from one IP address, if the administrator blocks the IP address exhibiting such malicious behavior, both abnormal and normal hosts behind the NATD will be damaged [3].

To operate the network effectively and securely, the administrator needs to verify whether the assigned IP addresses are using NAT or not. In particular, organizations operating large networks and addressing confidential resources such as documents, servers, and network facilities may prohibit unauthorized NATDs to prevent these problems. However, it is still a significant challenge to identify NATD in such large network operation environments.

Various studies to identify NATD have been carried out. They can be classified into two types: passive and active. In passive NATD identification methods, identifiers first sniff packets from an IP address and then decide whether it uses NAT by analyzing patterns of TCP/IP headers such as the identification [4] and time-to-live (TTL) [5] fields in the IP datagram header, timestamp [6], in the TCP header, and both IP and TCP header fields [7]. In [8], a method to detect NATDs and the NATHs behind them by using IP TTLs and HTTP user-agent strings is proposed. Methods to identify NATDs and NATHs using machine learning techniques based on packet sequences are proposed in [9–12]. Most of these passive methods presented their NATD identification methods as parts of techniques for counting the NATHs behind an NATD. Those passive methods need to monitor the sequence of packets from or to an NATD. However, the tasks for monitoring packets incur very high computational complexity and processing time. In addition, the identifier must be located on the same LAN segment as the NATD, to sniff packets. Passive identification at a remote location is not applicable.

Very few works on active NATD identification methods have been undertaken. In the active methods, the detector identifies an NATD by communicating with a target host or applications running on it behind the NATD [13–17]. However, these methods are mostly performed with knowledge of the presence of the NATD. Furthermore, they are unsuitable for determining whether NATD is used or not for specific IP addresses.

In this paper, we propose a supervised learning-based *active* NATD identification method using port response patterns, which can be performed remotely. We also propose a fast and stealthy identification method to solve the problems with regard to time and security that supervised learning-based methods exhibit. The main contributions of the proposed method are as follows.

- The proposed method provides a robust identification of NATDs independent of algorithms because it operates based on *supervised learning*.
- The proposed method operates in an *active* manner. It sends probe packets to the target hosts and collects the responses from them.
- In order to reduce the computational complexity and to solve the security issue in the collection of port response patterns, we propose a *fast* and *stealthy* identification method using the decision tree (DT) classification model.
- The proposed method can operate *remotely*, unlike conventional methods that should operate in the same network as the targets.
- With the fast, stealthy, and remote features, we also recommend a few *practical use cases* for secure network operation and management of an organization utilizing the proposed method.
- The NATD identification and stealthy active scan methods proposed in this paper have a symmetry feature applicable to a single subnet, small and medium-sized organization-level networks, and the Internet.

The remainder of the paper is organized as follows. An overview of NAT and related work on NATD and NATH identification are presented in Section 2. In Section 3, the proposed supervised learning-based NATD classification using port response patterns is explained. The experimental results on the effectiveness and the unresolved problems of the proposed method are presented in Section 4. In Section 5, the proposed stealthy and fast NATD identification method is presented. It is an extension of the method described in Section 3. In Section 6, some discussions on practical use cases utilizing the proposed methods are presented. Finally, the conclusions of the study are presented in Section 7.

2. Background

2.1. Nat Overview

NAT [1] enables multiple hosts with private addresses in private networks to access the Internet with one public IP address. NAT-enabled routers (NATDs) such as home routers and WiFi access points have been popularly utilized in homes, offices, public places, etc. In general, a NATD has an interface with a public IP address, which is connected to the public Internet. It has another interface(s) for hosts with private addresses in their private network(s). As mentioned before, these hosts with private addresses behind the NATD are called NATHs. To support applications between the NATHs and hosts on the Internet, NATD utilizes the network address and port translation (NAPT) method. It maps port numbers and IP addresses between them whenever they send or receive packets between themselves.

The advantages of NAT are as follows. First, it can solve the IPv4 address exhaustion problem in the short-term. Second, it enhances security. That is, external hosts beyond an NATD cannot access the NATHs directly. They can access the NATHs only after receiving request packets from the NATHs or by using a technique such as port forwarding that is supported by the NATD. Similarly, NATHs can be protected from malicious behaviors inflicted directly by external hosts.

However, NAT may cause problems for network operators in managing the network safely and efficiently. Because numerous NATHs may distribute a large amount of traffic simultaneously through an NATD, significantly more traffic management burdens may be imposed on network operators. NATDs with insufficient security policies are more vulnerable. For example, vulnerable services opened through port forwarding may enable attackers to conveniently access the internal network, and NATHs may be used as zombies for distributed denial of service (DDoS) attacks. When many NATHs are connected to the private network of an NATD, it is inconvenient for external hosts or systems to identify the malicious behaviors by any of these or the NATHs exhibiting such behaviors [2,3,18].

2.2. Related Work

The identification of NATD and NATHs is a highly important practical issue for network administrators to ensure effective and secure network operations. Many works have been conducted on the identification of NATD and NATHs, which can be classified into two types: passive and active.

The passive methods identify NATD by analyzing patterns of sniffed TCP/IP header fields. In [4] and [5], passive methods using IP identification (IPID) and TTL fields, respectively, in datagram headers have been proposed. As a host increases IPID by one whenever it sends a new packet, NATD can be identified by monitoring the linearity of the increment pattern of IPID. A packet's TTL is decreased by one when it passes a router. When packets from the same IP address have different TTL values, the device with the address is determined as NATD. Kohno et al. [6] proposed a method implemented in three stages: (1) store the sequence of timestamp values in TCP header, (2) analyze its distribution, and (3) identify the IP address as a single host or NATD according to whether the distribution follows a normal distribution or not, respectively. Park et al. [7] proposed a method to enhance the identification accuracy by combining IP and TCP header fields.

In [8], an HTTP user-agent string is utilized for NATD identification. In [19], domain name system (DNS)-based NATD detection has been proposed. As DNS requests are typically sent to a fixed IP address of the resolver, the IP-ID field of DNS requests is incremental. By tracking values of IP-ID of DNS requests coming from a single device, NATD can be detected.

Methods to identify NATD and NATH by using machine learning (ML) techniques based on packet sequences have been proposed. Li et al. [9] proposed a method using support vector machine (SVM) algorithm. Herein, the number of transmitted and received TCP, UDP, DNS, SYN, FIN, and RST packets was selected as the features and recorded every 2 min. Their algorithm exhibited an accuracy of 75%. Abt et al. [10] proposed a C4.5 DT algorithm based on flow information such as the IP addresses and port numbers of sender and receiver, protocol, and number of bytes and packets exchanged on

an IP address. The method resulted in an accuracy of 89%. Gocken et al. [11] compared the two machine learning algorithms Naive Bayes and C4.5 DT, with flow information as the features for NATD identification. They used 76 types of flow information and demonstrated that C4.5 DT exhibited higher performance than Naive Bayes. Komarek [12] proposed an identification method based on linear SVM and logistic regression (LR) by collecting HTTP proxy logs with eight HTTP features such as IP address, user-agent string, web browser, OS, connection status, upload and download byte, and the number of HTTP requests. The identification accuracy is 68% for three NATHs and 96% for five or more NATHs. Zhang [20] also proposed methods to detect NATD using ML algorithms such as SVM, C4.5, and alternative decision tree on HTTP data set artificially generated by the authors of [12], and the host identification behind the detected NATD using TCP/IP headers. In [21], a method to detect vulnerable Internet of Things (IoT) devices connected behind a NATD by applying ML algorithms has been proposed.

To apply passive methods, the identifier is requested to be located in the same network as the NATDs because it has to sniff packets from/to the NATDs. Moreover, the identifier needs to collect, store, and analyze packets in real time, which consumes excessive amounts of resources. Therefore, these methods may not be practical for a large-scale network. In addition, the available passive methods may be limited depending on the system environment. IPID is infeasible on Linux and Windows 10 OSs because these OSs alter the IPID field dynamically for security. The timestamp option is not a default in Windows. HTTP-based methods can be used only when the hosts are connected to the Internet. Thus, it is not available in an intra-network environment. Furthermore, deep packet inspection to examine the HTTP information in the application layer is required.

In the active NATD identification method, a detector actively communicates with NATHs or NATDs. However, to our knowledge, no practical active method has been studied. Rather, methods to support peer-to-peer communication between NATHs and external hosts beyond NATD have been proposed, such as NAT traversal or hole punching techniques [13], MAC address relaying [14], NATH identification with proxy authentication [15], Vulnerable NATH and NATD identification using UPnP [16], and WebRTC [17].

3. Supervised Learning-Based Active Natd Identification Using Port Response Patterns

3.1. Port Response Patterns

A port (or port number) is a virtual and logical identifier to distinguish between different services that operate over transport protocols such as TCP and UDP. It is represented as a 16-bit unsigned integer, thus ranging from zero to 65,535. A port is opened when it is assigned to a specific service running and closed when not in use. Therefore, the status of a port depends on whether the service assigned to it operates or not. With port scanning techniques, the status of ports can be determined. A port scanner sends request packets to the target ports of a host and then determines if they are in use (or open) by analyzing the responses on the ports' status from the host, called the *port response pattern*.

There are six typical responses when a TCP-SYN packet is sent to a port of a host, as shown in Table 1 [22,23]. The six responses are presented as indexes, 0x00, 0x01, $\cdots$, 0x05, in Table 1. There are three representative states of the port: *open*, *closed*, and *filtered* [24]. The open state indicates that the port is open and actively accepting TCP connections. The closed state implies that it is accessible although no services are operating through it. As a port response pattern for TCP-SYN, it may receive TCP-SYN/ACK or RESET segments explicitly when the port is open or closed, respectively. The filtered state denotes that it cannot determine the exact status of the port because the TCP-SYN packet may not have reached the port because of firewalls, access rules, or others. For a port in the filtered state, the host may respond with an ICMP error message with type 3 and codes 1, 10, or 13; or may not reply.

Table 2 presents the responses when UDP packets are used for port scanning. Here too, there are six response indexes. When the port is open, the host may respond very occasionally with a UDP packet. Meanwhile, an ICMP error message with type 3 and code 3 is sent when the port is closed.

Furthermore, it may not reply or may respond with ICMP error messages with type 3 and codes 1, 10, or 13. In these cases, it is determined to be in filtered states.

Table 1. Typical response patterns on a TCP-SYN scan.

Index	Response	Protocol	State	Notes
0x00	SYN/ACK	TCP	open	TCP flag:0x12
0x01	RESET	TCP	closed	TCP flag:0x14
0x02	no response	-	filtered	-
0x03	type3,code13	ICMP	filtered	communication administratively prohibited
0x04	type3,code10	ICMP	filtered	host administratively prohibited
0x05	type3,code1	ICMP	filtered	host unreachable

Table 2. Typical response patterns on a UDP port scan.

Index	Response	Protocol	State	Notes
0x00	UDP response	UDP	open	-
0x01	type3,code 3	ICMP	closed	port unreachable
0x02	no response	-	open or filtered	-
0x03	type3,code13	ICMP	filtered	communication administratively prohibited
0x04	type3,code10	ICMP	filtered	host administratively prohibited
0x05	type3,code 1	ICMP	filtered	host unreachable

3.2. Architecture of Supervised Learning-Based Natd Identification

We found that most of the devices, including NATDs and NATHs, have asymmetric port response patterns depending on different manufacturers and different products. For example, several IPtime and DLink NATDs have UDP port 67 and UDP port 68 closed ("0x01 Port Unreachable"). Apple's NATD, e.g., Airport, responds "0x03 admin prohibited" on all ports. Non-NATDs generally have TCP ports 22, 23, 80 and 443 and UDP ports 137 and 5353 open, with high "0x01 RESET" and "0x01 Port Unreachable" response rates. However, as IP devices including NATDs and NATHs evolve and release rapidly, their port response patterns may continue to change over time. Therefore, as the primary approach for the NATD identification, we consider a machine learning-based classification model that has trained on massive port response patterns of NATD and non-NATD.

The system architecture of the proposed supervised Learning based NATD identification method is shown in Figure 1. Herein, the overall process is divided into two phases: training and identification.

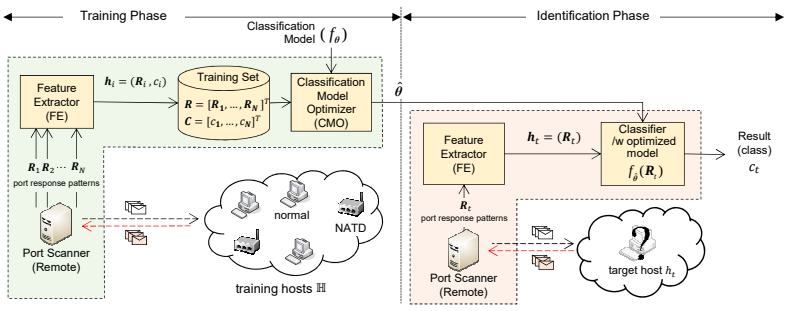

Figure 1. Architecture of supervised learning-based network address translation device (NATD) identification.

3.2.1. Training Phase

The training phase generates an optimal classification model $f_{\hat{\boldsymbol{\theta}}}$. Here, $\boldsymbol{\theta}$ is a set of parameters comprising multiple coefficients and a constant.

It is considered that there are N training hosts represented by a set of $\mathbb{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N\}$. Here, $\boldsymbol{h}_i$ denotes the i-th training host, $i = 1, \cdots, N$. Let $c_i \in \{true, false\}$ be the classified value for the i-th training host. It is known beforehand whether it is a NATD or not. That is, $c_i = true$ for a NATD, whereas $c_i = false$ for an ordinary host.

The training phase consists of three components: the port scanner, feature extractor (FE), and classification model optimizer (CMO), as shown in Figure 1.

The port scanner operates as follows: Let M be the number of predetermined ports to be scanned for each host. It sends M probe packets to all the target ports of a training host. This scan process is carried out repeatedly for all the training hosts. Then, M port response patterns are gathered from each host. Let $\boldsymbol{R}_i$ be the set of response patterns on M ports for the i-th training host. Then, we have

$$\boldsymbol{R}_i = \{r_{i,j} \mid j = 1, \cdots, M\}, \tag{1}$$

where $r_{i,j} \in \{0x00, 0x01, 0x02, 0x03, 0x04, 0x05\}$. Note that $r_{i,j}$ is one of the response indexes presented in Tables 1 and 2.

FE represents each host as a two-Tuple such that $\boldsymbol{h}_i \equiv (\boldsymbol{R}_i, c_i)$. Then, it combines all the hosts as a set of $\mathbb{H}$. Note that c_i is known prior to the training phase. Then, it extracts a feature set $\boldsymbol{R}$ and a class vector $\boldsymbol{C}$. Here, $\boldsymbol{R} = [\boldsymbol{R}_1, \boldsymbol{R}_2, \cdots, \boldsymbol{R}_N]^T$ and $\boldsymbol{C} = [c_1, c_2, \cdots, c_N]^T$. That is, $\boldsymbol{R}$ is denoted by an $N \times M$ matrix as follows.

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \vdots \\ \boldsymbol{R}_N \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,M} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{N,1} & r_{N,2} & \cdots & r_{N,M} \end{bmatrix} \tag{2}$$

The input to the CMO is $\boldsymbol{R}$ and $\boldsymbol{C}$ from FE. Let $f_{\boldsymbol{\theta}}$ be a classification model selected by the CMO. Any type of supervised learning-based classification models can be applied to this component. Then, CMO determines an optimal set of parameters $\hat{\boldsymbol{\theta}}$ as follows,

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} L\{f_{\boldsymbol{\theta}}(\boldsymbol{R}_i), c_i\}, \tag{3}$$

where L can be any of the loss functions depending on the classification model, e.g., Softmax loss function used in multilayer perceptron (MLP) [25]. Finally, the output from the CMO is an optimized classification model $f_{\hat{\boldsymbol{\theta}}}$.

The procedure for the training phase explained above is described in Algorithm 1. The operations of the port scanner, FE, and CMO are presented in lines 2, 3–4, and 6–8, respectively.

3.2.2. Identification Phase

The identification phase determines whether a target host ($\boldsymbol{h}_t$) is an NATD or not, without prior knowledge, i.e., $c_t \in \{true, false\}$. The phase utilizes the optimized classification model generated in the training phase, $f_{\hat{\boldsymbol{\theta}}}$. Three of its components are similar to three of the training phase: the port scanner, FE, and classifier.

The operations of the port scanner and FE are similar to those in the training phase. The operations in the identification phase are performed only for the target host, whereas they are performed for all the training hosts in the training phase.

The classifier applies the port response patterns $\boldsymbol{R}_t$ of the target host from the FE to its classification model $f_{\hat{\boldsymbol{\theta}}}$. Then, it generates a *Boolean* value c_t, *true* or $false$, indicating whether the target host is an NATD or not, respectively.

Algorithm 1 Training Phase

Input: Training Host Set $\mathbb{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N\}$ // $\boldsymbol{h}_i \equiv (\boldsymbol{R}_i, c_i)$
Output: Classification Model $f_{\hat{\theta}}$

```
for i = 1 to N do
    R_i = PortScan ( h_i );                    // Port Scanner (Equation (1))
    Append R_i to R ;                          // Feature Extractor (Equation (2))
    Append c_i to C ;
end for
repeat
    f_θ̂ = optimizeModel(f_θ̂, R, C);            // CMO (Equation (3))
until θ̂ converge
return f_θ̂
```

3.3. Selection of Target Ports

For the identification process, the port scanner sends probe packets to M ports of each host. The maximum value of M is 131,072, which is the sum of 65,536 and 65,536 (the number of TCP and UDP ports, respectively). The maximum M is very small compared to the number of features utilized in other ordinary machine learning applications, e.g., image processing. The processing time to learn and create the optimal model $f_{\hat{\theta}}$ from the port response patterns in the proposed identification method is lesser than those of other applications. However, it consumes a significant length of time to obtain the port response patterns by scanning those ports of all the training hosts. This will be discussed in detail in Section 5.1.

As not all ports affect the identification, the scan time can be reduced by selecting only some effective ones for the identification. *Nmap* [24] provides an open source scanning tool with a fast scan method that can be used for this purpose. The fast scan of *Nmap* scans 200 ports including well-known, registered, and some commonly used dynamic ports for efficient data collection. Therefore, the port scanner of the proposed method also utilizes *Nmap*'s fast scan method.

4. Evaluation

This section describes our evaluation of the feasibility of the proposed NATD classification by applying a few well-established supervised learning models. Owing to the non-availability of active-based identification methods, as mentioned in Section 2, only the proposed method is considered. For the evaluation, the dataset and comparable classification models that we use are first described. Next, we define metrics generally used in machine learning. Then, we present the validation of the proposed method and demonstrate which is the most effective classification model among the comparable models.

4.1. Dataset and Environment

We collected the datasets from five acting subnets operated in our university and access points (APs) open to the public so that we could identify whether each host was an NATD or not. We also collected a dataset from public access points outside our university that were using NAT. Table 3 shows the collected dataset with 264 NATDs and 302 normal hosts. With the dataset, we performed the Monte Carlo cross-validation. All samples were randomly divided into the training and test sets, and then the test was repeatedly performed a thousand of times to obtain the validated result. The size ratio of the training set to the test set has been set to 8:2. The proposed method has been implemented on a desktop computer with an Intel Core 2 i7-2600 and 16 GB of RAM using Python language, which is the fast and simple programming language and provides the wealth of machine learning libraries and frameworks. For the implementation, the Scikit-Learn package [26], one of the best known machine learning libraries, was utilized. The code and dataset for the evaluation are given at [27].

Table 3. Dataset for evaluation.

Area (C Class)	Total	NAT	Non-NAT
- . - . 196 . 0/24	31	20	11
- . - . 20 . 0/24	76	24	52
- . - . 197 . 0/24	114	26	88
- . - . 19 . 0/24	122	39	83
- . - . 21 . 0/24	167	99	68
public APs	56	56	0
Total	566	264	302

4.2. Classification Models and Metrics

As we illustrated in Section 2.2, existing methods focused on passive-based approaches. The main objectives of those conventional methods were to figure out counting NATHs and their behaviors behind NATD. Accordingly, as there are no appropriate conventional methods to compare to the proposed method, the following six typical supervised learning algorithms are used for comparing the algorithms for the application to NATD identification as well as the evaluation of the proposed method: Logistic Regression (LR) [28], Support Vector Machine (SVM) [29], K-Nearest Neighbors (KNN) [30], Multi-Layer Perceptron (MLP) [25], Decision Tree (DT) [31], and Random Forest (RF) [32].

LR is a probabilistic model in order to predict discrete outcomes in binary classification. Unlike linear regression, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can be mapped to classes. It is easy to implement and interpret, but there is a high probability that underfitting will occur.

SVM builds the hyperplanes as a classification boundary among the dataset by solving a quadratic programming problem. After training phase, the optimal hyperplane is selected. To overcome the low performance of linear classification, the original finite-dimensional space be mapped into a higher-dimensional space so that nonlinear classification is possible. It was the most powerful classification algorithm until the neural network became popular. SVM is not greatly influenced by noise and overfitting does not occur frequently. However, to train massive data, it requires more resources than other algorithms and it is difficult to understand how classification model were built and to analyze model.

KNN is an algorithm to classify with labels of adjustable parameter k, the number of nearest neighbors. The class is determined by a plurality vote of k neighbors. KNN does not required explicit training phase, but neighbors can be thought of as the training set for the algorithm. Because of neighbors can provide an explanation for the classification result, it is easier to understand than other black-box algorithms. However, the runtime speed is slowed down as the number of neighbors increases.

MLP is also known as an artificial neural network, which is an iterative optimization model that reduces the error rate by Feed-Forward and back propagation. It is a structure that connects several perceptrons which are linear classifiers and the gradient descent method calculates the optimal parameters of each perceptron. MLP is known as the most effective method for complex input such as computer vision domain, but it costs a lot of resources.

DT repeats the best split of data set according to classification criterion, resulting in a tree structure. The criterion for measuring the best split can be different for different algorithms. For example, ID3, C4.5, and C5.0 DT uses entropy-based metrics called information gain. The Classification and Regression Tree (CART) uses Gini impurity. DT is simple and fast. It is also a white box-based algorithm so that it is easy to find which conditions have affected the classification results. On the other hand, DT is a heuristic algorithm based on greedy algorithm, so it does not guarantee optimal tree generation.

RF is an algorithm to improve the disadvantages of DT. It creates one model by combining several DTs which are slightly different by randomness. Because it guarantees diversity, it is superior in

classification performance, but it cannot have intuition which is the advantage of DT in the analysis because it is a method which is a combination of multiple DTs.

The metrics used for the evaluations are precision, recall (also known as *sensitivity* or true positive rate (TPR)), accuracy, F1 score, and receiver operating characteristic (ROC) curves. These metrics are commonly used in machine learning approaches based on the confusion matrix presented in Table 4.

Table 4. Confusion matrix.

Prediction \ Type	NATD	Normal Host
NATD	TP (True Positive)	FP (False Positive)
normal host	FN (False Negative)	TN (True Negative)

They are defined as follows,

$$Precision = TP/(FP + FP). \tag{4}$$

$$Recall \ = \ TP \ / \ (TP + FN), \tag{5}$$

$$Accuracy \ = \ (TP + TN) \ / \ (TP + TN + FP + FN), \tag{6}$$

$$F_1 \ = \ 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{7}$$

The ROC curve is a probability curve. It plots the *sensitivity* of Equation (5) against the *specificity*, also known as the true negative rate (TNR) or the one-false positive rate, at various threshold settings. The *specificity* is defined as follows,

$$Specificity \ = \ TN \ / \ (TN + FP). \tag{8}$$

With an ROC curve, we can also have the area-under-the-curve (AuC) metric. It is an essential evaluation metric for examining the performance of a classification model. The closer the AuC is to one, the more effective the classification model is.

4.3. Performance Evaluation

In Table 5, the evaluation results (averages) of the performance metrics such as precision, recall, accuracy, F1 Score, and AuC for the different classification models are compared. As is evident from Table 5, LR and SVM exhibit the highest recall. However, they are lower than others in terms of the other metrics. The precision, accuracy, and F1 performances of KNN and MLP are higher than those of LR and SVM, albeit lower than DT and RF. From the observations, we can conclude that RF and DT exhibit relatively higher overall performances than the other models do. Because RF is an ensemble technique that combines several DTs, it performs marginally better than DT.

Figure 2 shows the ROC curves of the classification models. The AuC values obtained from the ROC curves for all the models, as illustrated in Table 5, are similar and are in the range between 0.94 and 0.97 (close to one). Moreover, the F1 scores for all the classification models are higher than 90%. In particular, the F1 scores of DT and RF are above 94%, significantly higher than those of the others. This implies that the proposed model based on port response patterns is highly effective for NATD identification.

Figure 3 shows the average elapsed time to train and test the classification models. KNN and DT exhibit the first and second fastest training times, respectively, among the algorithms. Note that the test time is more important than the training time. This is because in general, it is required to operate in real time, whereas the training phase is not. LR and DT exhibit highly similar and the fastest test times among the algorithms. As is evident from Table 5, LR exhibits significantly lower average precision,

accuracy, and F1 values than DT. Thus, DT is the most efficient algorithm considering both the test and training times.

Table 5. Performance comparisons of classification models.

	LR	SVM	KNN	MLP	DT	RF
Precision	88.4%	82.6%	90.4%	90.0%	92.1%	92.4%
Recall	97.5%	97.4%	95.2%	95.8%	96.7%	97.3%
Accuracy	92.9%	89.3%	93.1%	93.1%	94.7%	95.1%
F1	92.6%	89.2%	92.5%	92.7%	94.2%	94.7%
AuC	0.944	0.950	0.944	0.943	0.940	0.962

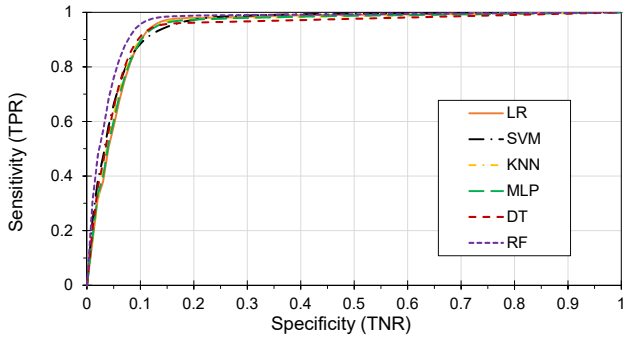

Figure 2. Receiver operating characteristic (ROC) curves for the classification models.

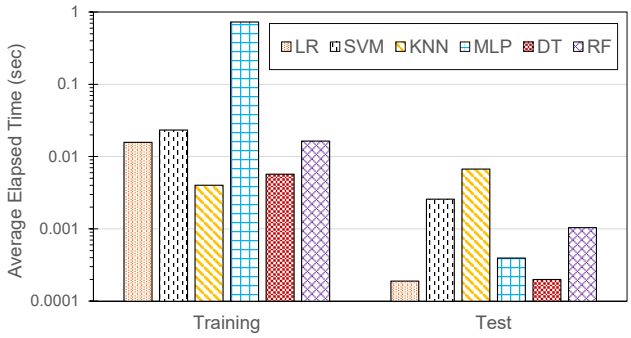

Figure 3. Average elapsed times for the training and test phases by classification models.

5. Decision Tree-Based Fast and Stealthy Natd Identification

5.1. Limitation of Supervised Learning-Based Approach

In Sections 3 and 4, we presented the proposed approach toward supervised learning-based NATD identification using port response patterns, and its evaluation. It exhibited very high effectiveness, with F1 scores of over 90%. Notwithstanding this high effectiveness, the supervised-learning approach has the following two main problems in practical applications.

First, it requires a very long time to collect port response patterns. It scans at a maximum of 65,536 ports for each of TCP and UDP, a total of 131,072 ports. To demonstrate the effect of the number of ports to be examined on the identification time, we configured two hosts as a scanner and a target. These were connected in the same subnet through an Ethernet switch with 1 ms latency. Then, the scanning on the ports was performed using *Nmap* [24] rather than the proposed system to provide a more general insight. The round trip time (RTT) was measured as the difference between the time when the scanner sends a probe packet and the time when it receives the response from the target. Note that after collecting all the response patterns on all the target ports, the classification or the identification task followed. In addition, we measured the processing time of the classification or identification task.

Figure 4a shows the elapsed time results of the TCP ports by varying the number of target ports. The RTT increases linearly as the number of ports increases. The processing consumes approximately 2 s independent of the number of ports. The elapsed times for the UDP port scans also increases linearly as the number of target ports increases, as shown in Figure 4b. However, the degree of increase is significantly larger than that of TCP. A comparison of Figure 4a,b reveals that the time for the TCP scan is not high compared to that for UDP scan. The UDP port scan was measured to consume approximately 1 s per UDP probe packet. UDP scan consumes a significantly longer time than TCP does because UDP is a stateless and connectionless protocol. The host should wait for a possible reply that may be processed. Furthermore, although the time-out value is set to zero for waiting, Linux OS places a limit of one reply per second for ICMP messages. According to Table 2, a UDP port may respond with an ICMP packet while the probe packet is filtered. This is why we selected 200 TCP and UDP ports for the experiments described in Section 4.

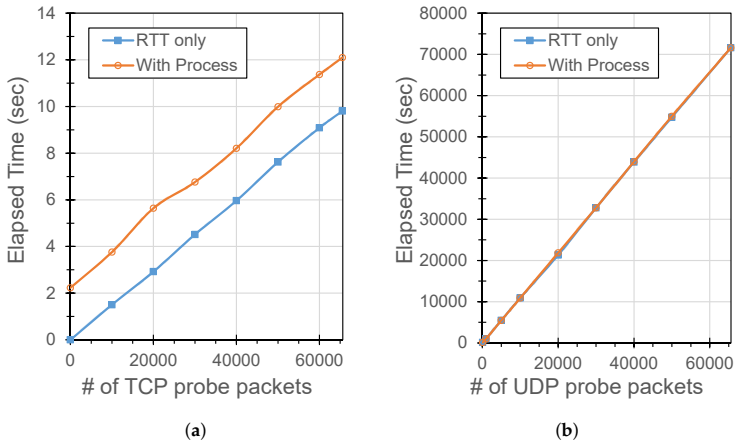

Figure 4. Average elapsed times to scan all targets (**a**) TCP and (**b**) UDP ports.

Second, intrusion detection systems (IDSs) and firewalls may consider probe packets as a threat or an attack. Because port scanning techniques have been typically utilized by malicious users to identify vulnerabilities of target hosts, IDSs and firewalls are generally configured to block the port scan packets as well as the host that sends the packets. To detect the port scan threat, a few studies and commercial services have defined their thresholds in the unit of packets per second (PPS). To detect TCP-SYN and UDP flooding attacks, threshold values were set to 20 incomplete TCP-SYN and 10 UDP PPS in [33] and 200 incomplete TCP-SYN and 300 UDP PPS in [34]. Commercial network devices such as routers and firewalls also have default rules unique to them to detect these attacks as follows, 128 incomplete TCP-SYN and 500 UDP PPS for Cisco [35] and 25 incomplete TCP-SYN PPS in the Juniper Networks firewall [36]. The threshold values are summarized in Figure 5.

The port scanner of the proposed system architecture described in Section 3 sends 100 TCP-SYN and 100 UDP PPS per host to collect its port response patterns. When there are numerous hosts to be scanned, PPS values increases tremendously and exceeds the thresholds shown in Figure 5. Then, the probe packets are all filtered by IDSs or firewalls.

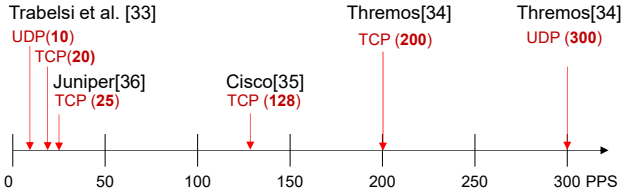

Figure 5. Default packets per second (PPS) values of intrusion detection systems (IDSs) and firewalls to detect TCP-SYN and UDP flooding attacks.

5.2. Dt-Fs: Dt-Based Fast and Stealthy Natd Identification

A simple method to solve the limitation illustrated in the previous subsection is to send fewer probe packets for a time period long enough to evade detection by IDSs or firewalls. However, this time period becomes excessively long as the number of hosts increases, which may be impractical. There have been some works to avoid the situation by optimizing the scanning rate considering congestions and throughput in wireless LAN environments [37], IP addresses and their collected features for Internet-wide scan [38], and so on. However, the scan target for those existing methods are not applicable to NATD identification. Therefore, another effective method is required to solve the problem.

Here, we propose a fast and stealthy NATD identification method based on DT (called *DT-FS*), to overcome the limitations of the supervised learning-based approach. With the evaluation results in Section 4.3, we select DT as the fundamental classification model. This is because it exhibited relatively higher performances, with 94.7% accuracy and 94.2% F1 score and lower elapsed times for the training and identification phases, than those of the others. In addition, the DT constructed in the training phase exhibits a hierarchical tree structure. Herein, the NATD identification rules are represented by the paths from the root to the leaf nodes. The identification process is repeated until it reaches the leaf node starting from the root. Similarly, because it does not visit all the nodes within the tree, we can achieve fast identification by inducing the port scanner to send probe packets only to the ports that it will visit on the DT's path.

The algorithm for DT-FS is presented in Algorithm 2. Each node $\mathbf{z}$ is represented as the feature vector $[c, m, r, \mathbf{z}_c]$, where $c \in \{true, false, TBD\}$ is the outcome at the node. In general, c at the root and intermediate nodes is set to *TBD*. It denotes *To Be Determined*. Meanwhile, c at the leaf node is *true* or *false*. m and r are the port number to be probed and its response pattern, respectively. $\mathbf{z}_c = \{\mathbf{z}_c(r) | \ r = 0x00, 0x01, \cdots, 0x05\}$ is the set of child nodes that it can be branched into. Here, $\mathbf{z}_c(r)$ is the child node to be branched for a response pattern r.

Algorithm 2 functions as follows. First, it examines c in $\mathbf{z}$. If c is *TBD*, it implies that the node is not a leaf node and needs more response pattern. Then, it sends a probe packet to the port m of the host h and obtains the response pattern r from h by using the function *SendProbePacket*( ). Here, $r \in \{0x00, 0x01, \cdots, 0x05\}$ (lines 1–2). Then, it branches to $\mathbf{z}_c(r)$, which is the child node to be branched by r among those listed in $\mathbf{z}_c$, and repeats the function *DT-FS*( ) (line 3). If c is *true* or *false*, it implies that the node is a leaf, and it returns c as the identification value (line 5).

Because DT-FS is based on DT as its classification model, its performances with regard to precision, recall, accuracy, F1 score, and AuC are identical to those of the supervised learning-based NATD identification method with the DT model (*DT-SL*) presented in Table 5.

However, unlike DT-SL, which collects all the response patterns on all the target ports of a host, DT-FS sends probe packets only to the ports that it will visit on the DT's path. Accordingly, DT-FS can reduce the time for NATD identification significantly, and that too with a substantially smaller number of port response patterns than that for DT-SL.

Algorithm 2 DT-FS Algorithm

Input: $\mathbf{z}$ and h // $\mathbf{z} = [c, m, r, \mathbf{z}_c]$, h is a target host
Output: boolean value // *true* or *false*
function *DT-FS* $(\mathbf{z}, h)$:
1: **if** c == *TBD* **then**
2: r = *SendProbePacket* (m, h); // get port response
3: *DT-FS* ($\mathbf{z}_c(r)$, h); // branch to child node
4: **else if** c == *true* OR *false* **then**
5: **return** c; // identification result
6: **end if**
end function

Figure 6 shows the cumulative distribution function for the number of probe packets required for the NATD identification. Here, γ_{train} and γ_{test} denote the ratio of the training set and of the test set, respectively, to the total dataset. Note that $\gamma_{train} + \gamma_{test} = 1$. As is evident from Figure 6, the identification can be conducted with a very small number of probe packets for a few hosts. For example, 40% of the hosts can be identified by at most five probe packets. With less than 30 probe packets, all the hosts are identified for all the cases of γ_{train} and γ_{test}.

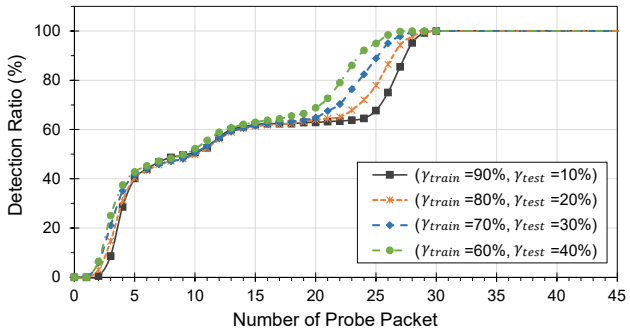

Figure 6. Detection ratio of decision tree (DT)-based fast algorithm by varying the number of probe packets for various dataset environments.

Furthermore, it is evident that the larger γ_{train} is, the higher the number of probe packets that need to be sent. This is because as γ_{train} increases, the paths from the root to the leaf nodes become deeper, and it requests more port response patterns. However, for all the cases, the tree depth for DT-FS can be optimized by a maximum of 30 probe packets.

With the result shown in Figure 6, NATD identification for a host can be performed with a small number of probe packets. Then, we can conveniently compute the elapsed time for identifying all the hosts in the networks managed by an IDS or firewall for its network protection as follows. Let Δ and K_{th} be the time unit and the threshold value of the number of packets for the detection of abnormal behaviors by an IDS or a firewall, respectively. Let M and N be the number of probe packets and hosts, respectively, for the NATD identification. Then, we have the total time of NATD identification without being detected by the IDS or firewall as $T_{NATD} = N \cdot \Delta \cdot \lceil M/K_{th} \rceil$. Here, $\lceil x \rceil$ denotes the smallest integer larger than x. For example, let us consider $M = 30$ and $N = 1$. Then, it requires 1 s for [35,36], and 2 s for [33,34], as shown in Figure 5.

6. Discussion: Practical Use Cases

6.1. Remote Natd Identification for Organization Networks

As mentioned earlier, whereas NAT provides various advantages, it may cause potential threats to network operations. To operate the network effectively and securely, the network administrator may have to verify whether an assigned IP address is using NAT or not. Conventional methods for the NATD identification have been based on passive ways focusing on detecting behaviors of NATHs behind a NATD, as shown in Figure 7 and proved their effectiveness. However, the NATD behavior detection should be done in the same subnet where the NATD is connected to, and its IP address is given in advance. Alternatively, the IP address of the NATD may be determined by capturing and storing all packets to and from all terminals in the subnet and analyzing the behaviors associated with each IP address. It may cause too big memory and computational burdens. To apply these passive methods, the network administrator must perform NATD identifications separately to individual subnetworks managed by it. This is highly complicated and unpractical for medium- or large-sized networks. In particular, as NATD is used as an IoT gateway these days, many NATDs will be installed in buildings [39,40]. Virtual NATDs implemented with virtual environment technology such as Docker [41] can also burden on network management.

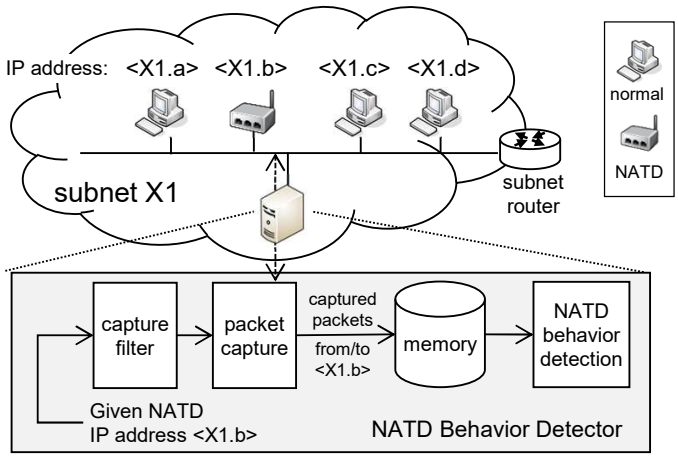

Figure 7. Passive NATD behavior detection by conventional methods.

Meanwhile, the proposed method for the NATD identification can perform on networks remotely to which the target IP address does not belong. The process by the proposed method is illustrated in Figure 8 and operates as follows. The network administrator installs a stealthy port scanner for NATD identification, selects target networks or hosts for scanning. Then, it scans all the hosts in the selected networks or the target hosts to collect their port response patterns. As mentioned in Section 5.2, the scan process can be performed remotely, stealthy, and fast. With the collected port response patterns, it can be determined whether each IP address uses NAT or not.

In this manner, the network administrator can classify whether or not IP addresses of all the networks he/she manages use NAT. Then, the IP addresses using NATDs may be stored in a database for their further management. Because NATDs with new port response patterns may emerge, it is also necessary to reconfigure the dataset for training and to update the CMO periodically.

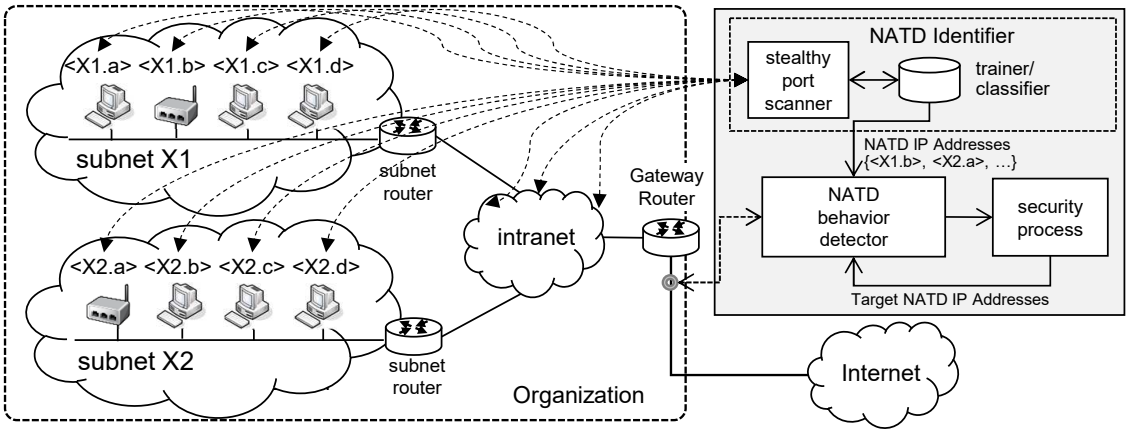

Figure 8. Use of proposed method in an organization.

6.2. Counting NATHs behind NATDs and Detecting Their Abnormal Behaviors

In order to ensure the stable and secure operation of an organizations' networks, it may be necessary not only to estimate the number of NATHs behind each NATD, but also to detect their abnormal behaviors. Numerous works have been undertaken to resolve the issue based on the analysis of the information of headers and payloads of packets captured from a NATD [4,8,42,43]. However, these conventional methods can be operated only with prior knowledge of the IP address of the NATD. Because the proposed method can be performed remotely, it is feasible to manage all the IP addresses of NATDs in an integrated manner. For example, the NATD identifier shown in Figure 8 acquires fast and stealthy all the IP addresses of the NATDs in an organization, and provides the

addresses to the NATD behavior detector. The NATD behavior detector can capture packets from and to the identified NATDs remotely at a gateway point connected to the external network, estimate the number of NATHs behind the NATDs, and analyze the behavior of NATHs. With the collaboration of the security process, it can also detect and control NATDs with NATHs doing malicious behaviors behind them.

7. Conclusions

In this paper, we proposed a supervised learning-based active NATD identification method using port response patterns. The proposed method can perform the identification remotely, unlike conventional methods that should operate in the same network as the targets. Experiments were conducted on approximately 600 samples with various feasible classification models. The experimental results demonstrated the effectiveness and feasibility of the proposed method, which exhibited an F1 score of over 90%. Comparisons among the various models revealed the DT model's relatively high performances. In addition, to reduce the time and to solve the security issue in the collection of port response patterns, we proposed a fast and stealthy identification method using the DT classification model, called DT-FS. It can identify NATDs by sending less than 30 probe packets.

We have recommended a few practical use cases of the proposed method. Traditional NATD identification schemes can be performed for devices connected in the same subnet due to their passive natures. Instead, the proposed method can detect NDTDs within an organization domain network via a remote active scan. We showed that the proposed scan could work stealthy and fast, not affecting internal firewalls and subnets within the domain. After NATDs have been identified, the behavior and status of NATHs behind each NATD can be investigated by monitoring packets with IP addresses for NATDs, not capturing all packets as in conventional methods. Though we discussed the practical use cases within an organization domain, it can be extended to the Internet-wide applications as in [38]. They are likely to contribute to secure and effective management of networks operating in organizations.

Author Contributions: Conceptualization, S.L. and B.-h.R.; methodology, S.L. and B.-h.R.; software, S.L.; validation, S.L., J.L., S.J.K. and B.-h.R.; formal analysis, S.L., J.L., S.J.K. and B.-h.R.; investigation, S.L. and B.-h.R.; resources, S.L.; data curation, S.L.; writing original draft preparation, S.L. and B.-h.R.; writing review and editing, S.L., J.L., S.J.K., and B.-h.R.; visualization, S.L. and B.-h.R.; supervision, B.-h.R.; project administration, B.-h.R.; funding acquisition, B.-h.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2018-0-01431) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). Furthermore, this work was supported by the Ajou University research fund.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Srisuresh, P.; Holdrege, M. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, IETF. 1999. Available online: https://www.hjp.at/doc/rfc/rfc2663.html (accessed on 10 October 2019).
2. Smith, M.; Hunt, R. Network security using NAT and NAPT. In Proceedings of the ICON'2002, Singapore, 27–30 August 2002; IEEE: Piscataway, NJ, USA, 2002.
3. Wicherski, G.; Weingarten, F.; Meyer, U. IP agnostic real-time traffic filtering and host identification using TCP timestamps. In Proceedings of the LCN'2013, Sydney, Australia, 21–24 October 2013; IEEE: Piscataway, NJ, USA, 2013.
4. Bellovin, S.M. A technique for counting NATted hosts. In Proceedings of the IMW'2002, Marseille, France, 6–8 November 2002; ACM: New York, NY, USA, 2002.
5. Phaal, P. Detecting NAT Devices Using sFlow. sFlow.org. 2009. Available online: https://ci.nii.ac.jp/naid/10019397892/ (accessed on 10 October 2019).
6. Kohno, T.; Broido, A.; Claffy, K.C. Remote physical device fingerprinting. *IEEE Trans. Dependable Secur. Comput.* **2005**, *2*, 93–108. [CrossRef]

7. Park, H.; Shin, S.; Roh, B.; Lee, C. Identification of hosts behind a NAT device utilizing multiple fields of IP and TCP. In Proceedings of the ICTC'2016, Jeju Island, Korea, 19–21 October 2016; IEEE: Piscataway, NJ, USA, 2016.
8. Maier, G.; Schneider, F.; Feldmann, A. Advertising power consumption of bluetooth low energy systems. In *Proceedings of the PAM'2011*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6519, pp. 32–41.
9. Li, R.; Zhu, H.; Xin, Y.; Yang, Y.; Wang, C. Remote NAT Detect Algorithm Based on Support Vector Machine. In Proceedings of the ICIES'2009, Wuhan, China, 19–20 December 2009; IEEE: Piscataway, NJ, USA, 2009.
10. Abt, S.; Dietz, C.; Baier, H.; Petrović, S. Passive remote source NAT detection using behavior statistics derived from netflow. In Proceedings of the AIMS'2013, UPC Barcelona, Spain, 25–28 June 2013; IFIP: Amsterdam, The Netherlands, 2013.
11. Gokcen, Y.; Foroushani, V.A.; Heywood, A. Can we identify NAT behavior by analyzing Traffic Flows? In Proceedings of the SPW'2014, San Jose, CA, USA, 17–18 May 2014; IEEE: Piscataway, NJ, USA, 2014.
12. Komarek, T.; Grill, M.; Pevny, T. Passive NAT detection using HTTP access logs. In Proceedings of the WIFS'2016, Abu Dhabi, UAE, 4–7 December 2016; IEEE: Piscataway, NJ, USA, 2016.
13. Ford, B.; Srisuresh, P.; Kegel, D. Peer-to-Peer Communication Across Network Address Translators. In Proceedings of the USENIX Annual Technical Conference, Anaheim, CA, USA, 10–15 April 2005.
14. Murakami, R.; Yamai, N.; Okayama, K. A MAC-address Relaying NAT Router for PC Identification from Outside of a LAN. In Proceedings of the SAINT'2010, Seoul, Korea, 19–23 July 2010; IEEE: Piscataway, NJ, USA, 2010.
15. Ishikawa, Y.; Yamai, N.; Okayama, K.; Nakamura, M. An identification method of PCs behind NAT router with proxy authentication on HTTP communication. In Proceedings of the SAINT'2011, Munich, Bavaria, Germany, 18–21 July 2011; IEEE: Piscataway, NJ, USA, 2011.
16. Rytilahti, T.; Holz, T. On Using Application-Layer Middlebox Protocols for Peeking Behind NAT Gateways. In Proceedings of the Network and Distributed System Security Symposium (NDSS) 2020, San Diego, CA, USA, 23–26 February 2020.
17. Cox, J.H.; Clark, R.; Owen, H. Leveraging SDN and WebRTC for Rogue Access Point Security. *IEEE Trans. Netw. Serv. Manag.* **2017**, *14*, 756–770. [CrossRef]
18. Vratonjic, N.; Huguenin, K.; Bindschaedler, V.; Hubaux, J. A Location-Privacy Threat Stemming from the Use of Shared Public IP Addresses. *IEEE Trans. Mob. Comput.* **2014**, *13*, 2445–2457. [CrossRef]
19. Orevi, L.; Herzberg, A.; Zlatokrilov, H. DNS-DNS: DNS-Based De-NAT Scheme. In Proceedings of the Cryptology and Network Security (CANS 2018), Naples, Italy, 30 September–3 October 2018; Springer: Cham, Switzerland, 2018; Volume 11124, pp. 69–88.
20. Zhang, L. Exploring NAT Detection and Host Identification. Master's Thesis, Dalhousie University, Halifax, NS, Canada, 2018.
21. Meidan, Y.; Sachidananda, V.; Elovici, Y.; Shabtai, A. Privacy-Preserving Detection of IoT Devices Connected Behind a NAT in a Smart Home Setup. *arXiv* **2019**, arXiv:1905.13430. Available online: https://arxiv.org/abs/1905.13430 (accessed on 15 May 2020).
22. Beverly, R. A robust classifier for passive TCP/IP fingerprinting. In Proceedings of the PAM'2004, Antibes Juan-les-Pins, France, 19–20 April 2004.
23. Postel, J. Internet Control Message Protocol. RFC 792, IETF. 1981. Available online: https://www.hjp.at/doc/rfc/rfc792.html (accessed on 10 October 2019).
24. Lyon, G.F. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*; Insecure: Sunnyvale, CA, USA, 2009.
25. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. *Learning Internal Representations by Error Propagation*; Technical Report; California Univ San Diego La Jolla Inst for Cognitive Science: La Jolla, CA, USA, 1985.
26. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
27. SL Based NAT Identification. GitHub. Available online: https://github.com/combatreadiness/SL-based_NAT_identification (accessed on 15 August 2020).
28. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. A training algorithm for optimal margin classifiers. In Proceedings of the COLT'92, Pittsburgh, PA, USA, 27–29 July 1992; ACM: New York, NY, USA, 1992.

29. Fix, E.; Hodges, J.L., Jr. Discriminatory analysis. Nonparametric discrimination: Consistency properties. *Int. Stat. Rev.* **1989**, *57*, 238–247. [CrossRef]
30. Cover, T.M.; Hart, P.E. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
31. Quinlan, J.R. *C4.5: Programs for Machine Learning*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1993.
32. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
33. Trabelsi, Z.; Alketbi, L. Using network packet generators and snort rules for teaching denial of service attacks. In Proceedings of the ITiCSE'13, Canterbury, UK, 1–3 July 2013; ACM: New York, NY, USA, 2013.
34. Thermos, P.; Takanen, A. *Securing VoIP Networks: Threats, Vulnerabilities, and Countermeasures*; Pearson Education: London, UK, 2007.
35. Cisco. *Security Configuration Guide: Denial of Service Attack Prevention*; Cisco Systems, Inc.: San Jose, CA, USA, 2014.
36. Juniper Networks. *Attack Detection and Prevention Feature Guide for Security Devices*; Juniper Networks, Inc.: Sunnyvale, CA, USA, 2019.
37. Hashida, H.; Kawamoto, Y.; Kato, N. Efficient Delay-Based Internet-Wide Scanning Method for IoT Devices in Wireless LAN. *IEEE Internet Things J.* **2020**, *7*, 1364–1374. [CrossRef]
38. Kim, H.; Kim, T.; Jang, D. An Intelligent Improvement of Internet-Wide Scan Engine for Fast Discovery of Vulnerable IoT Devices. *Symmetry* **2018**, *10*, 151. [CrossRef]
39. Jung, Y.; Agulto, R. Integrated Management of Network Address Translation, Mobility and Security on the Blockchain Control Plane. *Sensors* **2019**, *20*, 69. [CrossRef]
40. Nugur, A.; Pipattanasomporn, M.; Kuzlu, M.; Rahman, S. Design and Development of an IoT Gateway for Smart Building Applications. *IEEE Internet Things J.* **2019**, *6*, 9020–9029. [CrossRef]
41. Amirante, A.; Romano, S.P. Container NATs and Session-Oriented Standards: Friends or Foe? *IEEE Internet Comput.* **2019**, *23*, 28–37. [CrossRef]
42. Tekeoglu, A.; Altiparmak, N.; Tosun, A.S. Approximating the number of active nodes behind a NAT device. In Proceedings of the ICCCN'2011, Maui, HI, USA, 31 July–4 August 2011; IEEE: Piscataway, NJ, USA, 2011.
43. Mongkolluksamee, S.; Fukuda, K.; Pongpaibool, P. Counting NATted hosts by observing TCP/IP field behaviors. In Proceedings of the ICC'2012, Ottawa, ON, Canada, 10–15 June 2012; IEEE: Piscataway, NJ, USA, 2012.