Passive NAT Detection Using HTTP Access Logs

Tomáš Komárek
Czech Technical University
Prague, Czech Republic
komartom@fel.cvut.cz

Martin Grill
Czech Technical University
Prague, Czech Republic
grill@agents.felk.cvut.cz

Tomáš Pevný
Czech Technical University
Prague, Czech Republic
tomas.pevny@agents.felk.cvut.cz

***Abstract*—Network devices performing Network Address Translation (NAT) overcome the problem of the deficit of IPv4 addresses as well as introduce a vulnerability to the network with possibly insecure configurations. Therefore detection of unauthorized NAT devices is an important task in the network security domain. In this paper, a novel passive NAT detection algorithm is proposed that identifies NAT devices in the network using statistical behavior analysis. We model behavior of network hosts using eight features extracted from HTTP access logs. These features are collected within consecutive non-overlapping time windows covering last 24 hours. To classify whether a host is a NAT device or an end host (non-NAT device) a pre-trained linear classifier is used. Since labeled data for training purposes is hard to obtain, we also propose a way how to generate the training data from unlabeled traffic logs. On the basis of our experimental evaluation, the detection algorithm outperforms the state-of-the-art solution represented by [3].**

***Index Terms*—Network security, NAT detection, HTTP access logs, Support Vector Machine (SVM)**

I. Introduction

Network Address Translation (NAT) [1] is a technique that allows a single network device to act as an agent between public (e.g., the Internet) and private (local) network. In this way, only a single public IP address is needed to represent an entire group of hosts. The NATed network then gives the impression of being just one network device from the public network perspective. NAT is mainly used to overcome the deficit of public IP addresses. However, there are other justifiable reasons to introduce a NAT device (or NAT software) at exit points between the private and public network; such as hiding inner network topology, providing anonymity, content filtering or monitoring network performance.

Unauthorized NAT devices in the network may present a significant network security threat. Such devices are not included within the scope of IT security and therefore do not need to meet network authentication requirements, compliance checking for the endpoints, or other security measures required by the network security policy. Consequently, these devices may provide unrestricted access to the network for the attackers. Wireless NATs (e.g., Wi-Fi routers) pose a particular security risk since they may allow unauthorized access to the network from considerable distance without wired connection. The unauthorized NATs are typically introduced into network by the network users that want to evade some of the security policies (forbidden Bring Your Own Device) or when network administrators do not readily provide sufficient Wi-Fi coverage. In such scenarios, the users may bring their own Wi-Fi access point into the office and connect it to the corporate network.

Network administrator should be aware of all devices that potentially perform NAT in the network. Moreover, the identification of NAT devices is also important for the purposes of advanced network behavior analysis [2] systems. These systems build statistical models of network host's behavior based on network traffic in order to detect modern network security threats. Any deviation from the model is then reported to the administrator as an anomaly that should be investigated. In these systems, the behavior of NAT devices has to be, however, modeled separately as the traffic generated by NAT is in fact a mixture of behaviors of multiple individual hosts connected to the NAT.

The above particularities motivate the goal of this paper, which is to detect NAT devices in the network via behavior modeling. The problem can be viewed as a binary classification problem where each host in the network is either a NAT device (denoted further as a positive sample), or an end host device (denoted as a negative sample). Since the recognition of NAT devices is not inherently clear (all hosts appear like end hosts to the network administrator) and simultaneously crucial from the above mentioned reasons, NAT detection is an important and challenging task in the network security domain [3].

We address the problem using a supervised learning approach, which is based on learning from already known samples. The central premise is that two classes of hosts (NAT and end host) differ statistically in a way how they utilize the web [3][4]. We use the HTTP access logs as produced by the web proxy servers, described in detail in Section III, as our source of data. The key assumption is that the NAT devices should generate significantly more traffic (as illustrated in Figure 2) with higher diversity than the individual end hosts as they are composed of multiple hosts.

The supervised approach requires labeled samples from both classes. However in practice, obtaining labeled data in network security domains is difficult, time consuming, and expensive [4]. Additionally, when such data are available, there is typically insufficient amount of already known NAT devices compared to end hosts which makes the problem heavily imbalanced. Therefore, we propose a novel method to generate labeled data well suited for the training of a NAT detector using unlabeled traffic captures.

The contribution of this paper is twofold:

978-1-5090-1138-4/16/$31.00 2016 IEEE

1) It is shown that unauthorized NAT devices in the networks can be detected passively using behavioral analysis of HTTP access logs. To this end, a robust method for representing individual network hosts based on statistics derivable from the logs is designed.
2) In the second place, an effective method for generating a sufficiently large and manifold labeled data set from unlabeled traffic captures is proposed.

The reminder of this paper is organized as follows. Section II discusses relevant prior work. Section III provides a basic background in HTTP access logs. The proposed NAT detection method is introduced in Section IV and the supporting procedure for generating labeled data is described in detail in Section V. Experimental evaluation using real network data can be found in Section VI. The final Section VII concludes the paper.

II. Prior Work

There are two main approaches to infer NAT devices in the network: active and passive. The active ones interact with hosts by sending and receiving data packets. The communication, however, might not be permitted in all network environments. Therefore we focus on passive detection techniques which unlike the active ones do not generate any network traffic by sending data packets. They try to identify NAT devices just by observing the network traffic. Moreover, these techniques are undetectable, non-intrusive and capable to analyze stored historical records of traffic retrospectively. On the other hand, they require a convenient monitoring point in the network to be able to capture communication of each host.

These techniques can be further categorized into two groups: signature oriented and behavior oriented.

A. Signature-Based Algorithms

Following methods are representatives of the signature based group as they seek for signatures in Transmission Control Protocol and Internet Protocol (TCP/IP) packet headers.

Bellovin [5] proposes a technique for detecting NATted hosts based on observing IP ID sequences. It uses the fact that some Operating Systems (OSs) (e.g., Windows and FreeBSD) implement this field as a linear congruential generator producing consecutive numbers. Thus, the count of observed sequences originating from a network device corresponds to the number of hosts behind the device. However, the technique does not work for OSs where the IP ID is generated in a random manner (MacOS, OpenBSD) or as a per flow counter (current Linux distributions). Additionally, if Dont Fragment bit (DF) is set to one, some devices reset IP ID to zero.

Fukuda [6] extends Bellovin's technique by observing patterns in another two fields: sequence number and source port. Nevertheless, it still fails to detect OpenBSD hosts, because they implement all the fields in a random manner.

Beverly's method [7] tries to infer host's OS (OS fingerprinting) using several values of packet header fields. The idea is that a host having more OSs might indicate a NAT device. This approach is not capable of identifying NATted hosts with the same OS that are common, e.g., in corporate networks. Furthermore, it gives false positives when a host uses more different OSs installed either on its native machine or in a virtual environment.

Maier [8] combines information from the packet header with Hypertext Transfer Protocol (HTTP) meta-data contained within the packet payload to build more robust method. Specifically, the information about host's OS and Internet browser as parsed from User-Agent string is used. The technique performs poorly when NATted hosts use the same OS and/or Internet browser. This situation is, however, common in networks with homogeneous OS/software installations (e.g., banking and financial industry, public administration and some corporate environments). The technique also might produce false alarms when a host switches to a different OS or upgrades its Internet browser in the time of measuring.

B. Behavior-Based Algorithms

Algorithms proposed by Rui [4] and Abt [3] collect statistics about the host behavior in the network. The statistics of several features are gathered within a time window of a predefined length. At the end of the collecting phase, every host is represented by its own feature vector. To be able to decide whether a host is a NAT device or an end host, a classifier is learned on a set of already labeled feature vectors. The richer the samples of both classes of hosts are, the better classifier is learned and applicable in the future on yet unseen samples. Hence the performance of these techniques primarily depends on the amount and quality of the provided data.

Rui [4] applied Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel to detect NAT devices and estimate the number of hosts behind them. The binary classifier is fed by eight-component feature vectors, which are constructed by capturing statistics derivable from NetFlow records [9] about the host traffic using 100 seconds windows. The hypothesis is that NAT devices send more bytes, produce more connections, use more protocols and exhibit more complex behavior than end hosts. To train the classifier, traffic of five hosts was captured, labeled and analyzed in a lab. The maximum achieved detection accuracy in the binary task NAT/end host is 85.18%. Their implementation also showed that the accuracy is higher when there are more hosts behind the NAT device.

Abt [3] reported the highest classification accuracy of 89.35% on a balanced dataset (i.e., the same amount of both classes). Their method relies on nine features extracted from NetFlow records [9] and uses the time window set to 120 seconds. The machine learning algorithm C4.5 is applied to build a decision tree classifier. The NetFlow records come from a real-world environment with a majority of business customers. They are provided and labeled by a German Internet service provider (ISP). Unfortunately, numbers of hosts nor sizes of NAT devices are not specified at all. Feature vectors produced by a small number of hosts are not able to capture all aspects of communication of that particular class of hosts. Referring to NAT sizes, detection of huge NATs with many

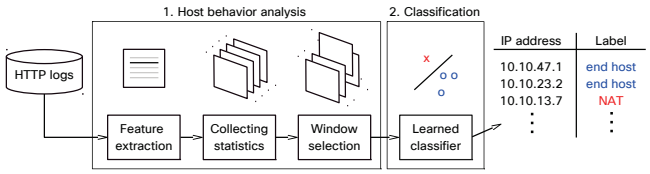

Fig. 1. Two stage detection procedure. The host behavior analysis generates behavioral features that are consequently used in the classification.

hosts connected to them is an easier task than a revelation of small NATs with a few host behind as already noticed by [4] and verified in Section VI-C by us. From the security point of view, the small devices performing NAT might, however, be more important. As they can allow unauthorized access to the network which is hard to discover. Additionally, the classifier is evaluated only on time-changing data coming from the same networks as the training data but a different period of time. We consider testing classifier's generalization ability on yet unseen networks to be more important property in the overall assessment (Section VI-B2).

III. HTTP Access Logs

The proposed NAT detection mechanism uses HTTP access logs as produced by the web proxy servers located on the network perimeter. Proxy servers are used by the network hosts to access web content. These are already present in many corporate networks to provide security and control over the web content downloaded by the network users.

The access logs contain information extracted from HTTP headers, each log entry represents one request made by a host to a web server. It contains host's IP address, the IP address of the server, Uniform Resource Locator (URL) of the request, the User-Agent information (identifying host's software originating the request), the sum of uploaded/downloaded bytes, the ending time of communication (timestamp) and its duration. Furthermore, there can be additional fields like the MIME type, HTTP status or method, etc. depending on the configuration of a particular proxy server.

IV. Detection Method

The proposed algorithm can be viewed as a two-stage procedure depicted in Figure 1. First, features that represent host's behavior are extracted from each entry of HTTP access logs and accumulated in consecutive non-overlapping time windows of a predefined interval covering last 24 hours. Secondly, a representative value of each feature is selected from the sequence of the windows to be used in the classification phase.

A. Host Behavior Analysis

For each host identified in the logs[1], the collection of eight features is extracted to differentiate NAT devices from end hosts:

1) the number of unique contacted IP addresses,
2) the number of unique User-Agent strings used in the HTTP requests,
3) the number of unique OSs including their versions[2],
4) the number of unique Internet browsers including their versions[3],
5) the number of persistent connections[4] [10],
6) the number of uploaded bytes,
7) the number of downloaded bytes,
8) the number of sent HTTP requests.

The features are collected in consecutive non-overlapping time windows with a predefined length. The sequence of windows (for each feature one) forms a first-in first-out (FIFO) queue covering last 24 hours. In our experiments, we set the period to 30 minutes as it yielded to slightly better results than 5, 15 or 60 minutes periods. This time span also proved to be the right balance between memory requirements and the computational cost. Since longer periods would require more memory to store the unique values of some features and shorter periods would induce more windows in the sequence among which it is needed to select the 90% quantile as described in the next paragraph[5].

Figure 2 shows an example of such sequence of the number of sent HTTP requests for both classes of hosts. The values represent an activity of an average host in the network during a day. They vary from low numbers in non-active night time up to maximal ones when the host exhibits an abnormal activity (e.g., downloading large files, automatic link checking, etc.). Both extremes of the host activity are misleading [4]. The non-activity tends the classifier to misclassify NAT devices with sleeping hosts and conversely the short-time abnormal activity to cause false positives in the case of end hosts. To make the detector robust to these events, only one non-extreme representative window for each feature is selected according to a predefined quantile. Our experiments showed that good results can be achieved by selecting 90% quantile for all features. Further tuning using feature selection methods did not bring any significant improvement. At the end of this stage, every host is represented by its feature vector consisting of eight components.

B. Classification

In the next stage, the host is classified as either a NAT device, or an end host device, based on its feature vector defined above.

We decided to employ the Support Vector Machine (SVM) supervised learning algorithm with the linear kernel to learn a decision rule from the training data. While the linear SVM is known for its simplicity and the speed of evaluation, it

[1]The host IP address is used as the unique identifier.

[2]OS names including their versions are parsed from User-Agent strings.

[3]Names of Internet browsers are parsed from User-Agent strings as well.

[4]A connection to an IP address is considered to be persistent if it is active in at least five time windows from ten last consecutive time windows. The windows are set to have the size of one, two and four hours.

[5]This observation is related to the datasets used in our experiments in Section VI.

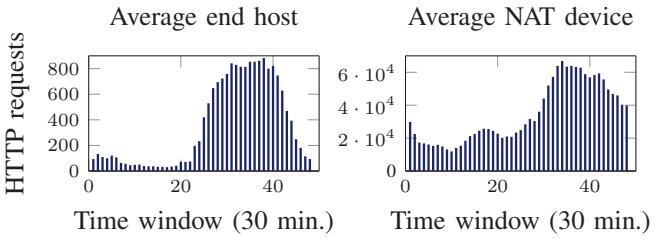

Fig. 2. A sequence of 30 minutes windows covering 24 hours that represents activity (in terms of sent HTTP requests) of an average end host (left) and an average NAT device (right) in the network. Note different scales of the vertical axes on the figures.

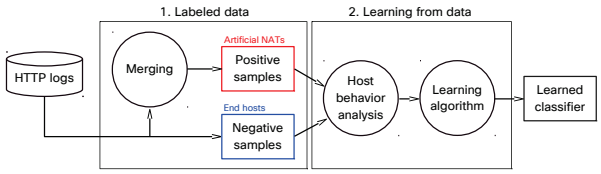

Fig. 3. Two stage learning procedure. During the first stage the labeled data is generated using the unlabeled HTTP log entries. All the entries are labeled as negative examples, all HTTP log entries of selected subset of the hosts are then duplicated and the host identifier is changed forming HTTP entries of a newly created artificial NAT host. In the second stage, the behavioral features are extracted and the classifier is trained as described in Section IV.

also showed to be immune to contaminated datasets compared to other commonly used algorithms [12]. The later property is beneficial in our case, since the process of generating the training data from unlabeled traffic might introduce wrongly labeled positive samples (more on this is in Section V-B).

V. Labeled Data

The efficacy of the proposed method for NAT detection, similarly to any other supervised method, depends heavily on the amount and quality of the training data. This exposes the classical problem of insufficient amount of labeled data in the network security domain where manual labeling is time consuming and in some cases even impossible.

Considering the NAT detection task, manual labeling is feasible only in the case of huge NAT devices (i.e., with many connected NATted hosts) as there is enough evidence in the HTTP logs. But, finding smaller NAT devices is much harder. On top of that the NAT detection problem is heavily imbalanced since the number of NAT devices is usually negligible when compared to the total number of all hosts. Knowledge of the inner network topology is helpful, but not sufficient as there might be unauthorized NAT devices.

A. Approaches Taken by Prior Art

Rui [4] created a synthetic dataset of the network traffic collected from five computers. We consider this dataset to be too small to capture the behavior of thousands of real network hosts in a typical network environment.

Abt [3] on the other hand used a dataset from a real network environment provided by German ISP. The dataset was labeled according to an expert knowledge of the sponsoring ISP. However, neither the numbers of hosts nor the sizes of NATs in terms of connected hosts are specified. Although the dataset is balanced in terms of labeled features vectors, the portions of individual hosts which produced the feature vectors are unknown. This could introduce a bias in the training favorable of these specific hosts/networks, which could not be discovered as the testing set originates from the same networks but only in a different period of time.

As can be seen from the previous works, labeled data with sufficiently large number of positive and negative samples, especially with a large variety of positive samples (NATs of various sizes) is really hard to obtain. We address this issue via the proposed method for generating artificial NATs, described below.

B. Generating Artificial NATs

Assuming that we have unlabeled data of a regular network with the majority of the hosts being end hosts (non-NATs), we can leverage the fact that NAT devices join traffic of multiple hosts into one. Using this we can create artificial NATs simply by joining existing log entries generated by multiple network hosts together. This is possible due to the fact that none of the features listed in Section IV-A are modified by NAT device during the packet translation process.

During the NAT generation process, all hosts of the network are first marked as negative (non-NAT) examples and then logs of several selected hosts are duplicated and merged to create one positive sample. Thus, sampling with replacement is used, but logs of a specific host can be used only once for a particular NAT device. In this way, it is possible to generate an arbitrary number of positive samples of various sizes. The process is illustrated in Figure 3.

The mislabeling error, introduced by marking all network hosts as end hosts (true NAT devices present in the network are mislabeled as negatives), is usually insignificant due to the heavily imbalanced property of the original data. According to [11] and our experiment in Section VI-B4, contaminated datasets with a few mislabeled samples do not represent a significant issue for the training process. In addition, we employed the linear SVM classifier, which showed to be resistant to mislabeled samples in the training set compared to other widely used classifiers as demonstrated by [12].

Without any prior knowledge of the distribution of NAT sizes, we generated artificial NATs uniformly from three connected hosts up to twelve. We assume that the smallest detectable NAT device consists of three hosts and that it is rare when a single host produces more HTTP requests than twelve randomly selected end hosts together. A priori probabilities of individual classes of hosts are also unknown and even can not be known in principle, because there might be networks with one huge NAT device and also with none at all. Accordingly, it is reasonable to suppose that each class is equally likely to be observed and generate a balanced dataset.

VI. Experimental Evaluation

A. Datasets Specification

In the first four experiments, we used anonymized HTTP access logs captured over the time period of two working days in four different corporate networks. The networks were selected to be of various sizes with the number of hosts varying from 3 000, 5 000, 10 000, up to 25 000. The network access log captures of these networks do not include any labels. To have the labeled data for training and testing purposes, we generated artificial NATs as described in Section V-B and merged the generated HTTP access log entries with the entries of already presented hosts in the networks.

In the last experiment, we used anonymized HTTP access logs from a corporate network counting 1717 network hosts out of which 166 are authorized NAT devices.

B. Experimental Evaluation

We have evaluated the proposed NAT detection method using the datasets described above in five separate scenarios to prove its robustness.

1) Time-drift: The classifier is trained on the data of all networks from the first dataset captured during the first day only. In this setup, samples from all networks are used in the training similarly to the approach taken by Abt [3]. Evaluation of the trained classifier on the data captured from the second day shows stability of the method over time.

The results from the top of Table I indicate that the detector trained on samples from one day is able to reliably operate on future days in the same networks. This is supported by the fact that there is no significant drop of the accuracy in between the training and testing.

2) Cross-validation: In the next scenario, the trained classifier is tested whether it is able to operate on a new, yet unseen, network. For this "leave-one-network-out" cross-validation approach is used. The classifier is trained on three networks and then evaluated on the remaining one. The cross-validation process is repeated four times, so that each network is used exactly once as the evaluation set.

Table I shows results of the described scenario. The minimal evaluation metric value achieved from the worst case combination of the training and testing data is shown. The results imply that the detector trained on samples from a few networks can be deployed on a new, yet unseen, network with only a minor loss of performance. We consider the accuracy 94.75% from this experiment to be the representative result, since it comes from the evaluation under the worst-case scenario on the balanced dataset. As the potentially mislabeled real NAT devices are among classifier's false positives (FPs) inaccurately, the result should be interpreted as a lower bound value.

3) Hosts with the same OS and/or Internet browser: Additionally to the experiments described above, we have evaluated two extreme scenarios with NATted hosts having the same OS and/or Internet browser by merging the appropriate active hosts from all networks. This is common in the majority of networks in the banking and financial industry, public administration and corporate environments where the default setup of computers is enforced. In these networks, OS/browser fingerprinting methods [7][8] unavoidably fail, since these two features alone are not able to separate the classes. Nevertheless, the proposed solution is capable of detecting NAT devices with a high accuracy and only a small drop in recall as can be seen in the bottom half of Table I.

TABLE I
Results from experiments

	accuracy	precision	recall
Time-drift			
Training set	95.81%	97.39%	94.14%
Testing set	95.52%	96.74%	94.21%
Cross-validation "leave-one-network-out"			
Worst-case	**94.75%**	94.44%	90.36%
Hosts with the same OS			
Testing set	93.91%	98.31%	89.35%
Hosts with the same OS and Internet browser			
Testing set	94.31%	89.21%	84.35%
Network with real NAT devices			
Testing set	99.36%	98.14%	95.18%

4) Sensitivity to contaminated training set: In this experiment, the impact of presence of real NATs in the unlabeled data on the final performance is put to the test. As the real NATs presented in the original data may pose a problem for training, since they will be wrongly labeled as negative samples in the phase of generating artificial NATs described in Section V-B. As result, the real NATs may confuse the learning algorithm. To mitigate the impact, we employed the linear SVM learning algorithm which showed to be resistant to contaminated training sets compared to other widely used algorithms [12]. Nevertheless, we decided to evaluate the sensitivity by modifying the above described time-drift scenario. In addition to training on one day and evaluating on another, we are now progressively introducing mislabeling error[6] by turning labels of positive samples to negative ones in the training set and measuring the impact on the accuracy of classification on the untouched testing set.

Figure 4 illustrates that the presence of real NATs in unlabeled data is not crucial. Even if around 10% to 20% of all hosts in the network were in fact wrongly labeled real NATs, the impact on the learning process is insignificant.

5) Evaluation on the network with real NATs: To verify whether the classifier learned by the proposed method (including only artificial NATs) has indeed the capacity to detect

[6]In reality, we are gradually adding more mislabeled NATs to the already possibly presented real NATs in our data.

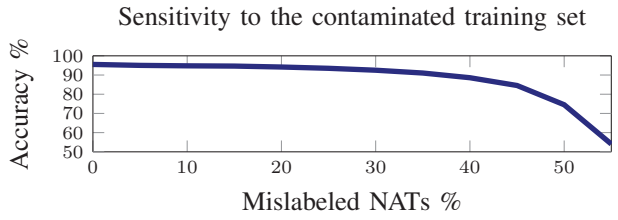

Fig. 4. Influence of the number of mislabeled NATs in the training set on the accuracy in the time-drift scenario.

real NATs in a previously unseen environment, we conducted the last experiment. The already trained detector from the cross-validation experiment (Section VI-B2) is used to detect NAT devices in the second (i.e., fully labeled) dataset. The labels were assigned using a list of network's authorized NAT devices. Although not necessarily complete, this list can give us an insight of how the proposed method works in practice.

As shown in the last row of Table I, the detector correctly identified 95.18% of authorized NATs with the precision 98.14%. In more detail, there were 3 FPs (i.e., possibly unauthorized NATs) out of 1551 end hosts and 158 correctly identified NATs out of 166 authorized NAT devices. Hence training on the labeled data, which are hard to obtain in this domain, can be substituted with the training on the generated positive samples (i.e., artificial NATs) and unlabeled samples from a regular network.

C. Analysis of FNs

An advantage of the proposed approach with artificial NATs is that the distribution of NATs is under our control. Hence the detector can be easily trained for specific network environments (e.g., the ratio of classes, sizes of NAT devices, NATted hosts having the same OS or Internet browser, etc.). In our experiments, the classifier is trained for a general case with the balanced ratio of hosts and the uniform distribution of NAT sizes ranging from three up to twelve connected hosts. Figure 5 shows an error rate of misclassified NATs from the first experiment related to the size of NAT device.

It is observed that the detector missed 37.3% of all NAT devices of size three. However, the error rate decreases exponentially and NAT devices with five connected hosts are already detected in more than 96% cases. From this illustration, it is clear that huge NAT devices with dozens of hosts are easily detectable when compared to small ones with a few (i.e., three or four) NATed hosts.

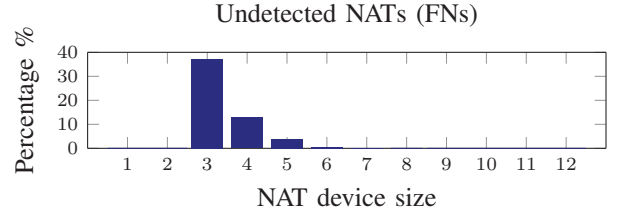

Fig. 5. Error rate from Time-drift experiment. While NAT devices with three connected NATted hosts are detected only in 62.7% of cases, NAT devices of the size five are already detected in more than 96% of cases.

VII. Conclusion

The paper presents a novel passive NAT detection algorithm, which is based on behavioral analysis of network hosts. Unlike the previous works [4][3], HTTP access logs are used instead of NetFlows as source of data. The representation of the network host is designed to be robust to short-time abnormal changes in behavior by selecting 90% quantile for each of eight features. The host is classified as either a NAT device or an end host, using the pre-trained linear SVM. Since the classifier needs a lot of already known samples to be well trained, which is hard to obtain in this domain, we propose a way to generate it from unlabeled traffic logs of a network. We demonstrated that the network containing 20% or less of real NATs can be used. This assumption is well satisfied for the majority of regular networks.

Acknowledgment

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

References

[1] P. Srisuresh and K. Egevang, "Traditional ip network address translator (traditional nat)," Tech. Rep., 2000.

[2] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," *NIST special publication*, vol. 800, no. 2007, p. 94, 2007.

[3] S. Abt, C. Dietz, H. Baier, and S. Petrović, "Passive remote source nat detection using behavior statistics derived from netflow," in *Emerging Management Mechanisms for the Future Internet*. Springer, 2013, pp. 148–159.

[4] L. Rui, Z. Hongliang, X. Yang, X. Yang, and W. Cong, "Remote nat detect algorithm based on support vector machine," in *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*. IEEE, 2009, pp. 1–4.

[5] S. M. Bellovin, "A technique for counting natted hosts," in *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, ser. IMW '02. New York, NY, USA: ACM, 2002, pp. 267–272. [Online]. Available: http://doi.acm.org/10.1145/637201.637243

[6] S. Mongkolluksamee, K. Fukuda, and P. Pongpaibool, "Counting natted hosts by observing tcp/ip field behaviors." in *ICC*. IEEE, 2012, pp. 1265–1270.

[7] R. Beverly, "A Robust Classifier for Passive TCP/IP Fingerprinting," in *Proceedings of the 5th Passive and Active Measurement (PAM) Workshop*, Apr. 2004, pp. 158–167.

[8] G. Maier, F. Schneider, and A. Feldmann, "Nat usage in residential broadband networks," in *Passive and Active Measurement*, ser. Lecture Notes in Computer Science, N. Spring and G. Riley, Eds. Springer Berlin Heidelberg, 2011, vol. 6579, pp. 32–41.

[9] "RFC 3954 - Cisco Systems NetFlow Services Export Version 9," Oct. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3954.txt

[10] J. Jusko, M. Rehak, and T. Pevny, "A memory efficient privacy preserving representation of connection graphs," in *Proceedings of the 1st International Workshop on Agents and CyberSecurity*. ACM, 2014, p. 4.

[11] G. Blanchard, G. Lee, and C. Scott, "Semi-supervised novelty detection," *J. Mach. Learn. Res.*, vol. 11, pp. 2973–3009, Dec. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1953028

[12] L. Bruzzone and C. Persello, "A novel context-sensitive semisupervised svm classifier robust to mislabeled training samples," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 47, no. 7, pp. 2142–2154, July 2009.