Passive Remote Source NAT Detection Using Behavior Statistics Derived from NetFlow

Sebastian Abt[1], Christian Dietz[1], Harald Baier[1], and Slobodan Petrović[2]

[1] da/sec – Biometrics and Internet Security Research Group
Hochschule Darmstadt, Darmstadt, Germany
{sebastian.abt,christian.dietz,harald.baier}@h-da.de
[2] Norwegian Information Security Laboratory
Gjøvik University College, Gjøvik, Norway
slobodan.petrovic@hig.no

Abstract. Network Address Translation (NAT) is a technique commonly employed in today's computer networks. NAT allows multiple devices to hide behind a single IP address. From a network management and security point of view, NAT may not be desirable or permitted as it allows rogue and unattended network access. In order to detect rogue NAT devices, we propose a novel passive remote source NAT detection approach based on behavior statistics derived from NetFlow. Our approach utilizes 9 distinct features that can directly be derived from NetFlow records. Furthermore, our approach does not require IP address information, but is capable of operating on anonymous identifiers. Hence, our approach is very privacy friendly. Our approach requires only a 120 seconds sample of NetFlow records to detect NAT traffic within the sample with a lower-bound accuracy of 89.35%. Furthermore, our approach is capable of operating in real-time.

Keywords: Network Address Translation, NAT detection, NetFlow, C4.5, SVM.

1 Introduction

Network Address Translation (NAT) [1] is a technique commonly employed in computer networks to hide a number n of computing resources behind a, typically smaller, number $m \leq n$ of IP address resources. The motivation for deploying NAT nowadays is twofold: (i) IP address conservation and (ii) security. The first motivation comes with the continuing growth of the number of Internet connected devices and, consequently, the continuous depletion of public IP version 4 address resources. The second motivation is based on the assumption that utilizing NAT efficiently hides internal network structures to the outer world. In fact, directly connecting to a host behind a NAT gateway from outside the NAT network is typically not possible.

From a network management and network security point of view, NAT may not always be desirable or permitted: enterprise networks typically enforce strict

G. Doyen et al. (Eds.): AIMS 2013, LNCS 7943, pp. 148–159, 2013.

security and management policies on connected devices in order to prohibit unwanted or illegal use of computing or network resources and to limit spread of malware. Such policies can be circumvented by connecting private devices to an enterprise network via NAT. Additionally, telecom operators and Internet Service Providers (ISPs) may prescribe bounds on the number of devices concurrently connected to a single Internet access point (e.g. 3G/4G connection sharing). Such limitations can easily be circumvented by connecting a NAT gateway to the Internet access point and covering further hosts behind this gateway.

In order to be able to identify such security and policy violations, we present a novel approach to remote source NAT detection by applying machine-learning algorithms to user behavior statistics derived from NetFlow [2] data. Our approach works completely passively, i.e. it does not require interaction with the monitored hosts. It can be implemented at any position within a computer network and is capable of detecting NAT with just a 120 seconds sample of NetFlow records with a lower-bound accuracy of 89.35%, outperforming existing related work. Furthermore, our approach respects privacy of end-users, which is especially important for enterprise networks: as our approach is completely based on information derived from NetFlow data, no payload has to be processed. Additionally, our approach does not rely on original IP addresses. Instead, it is capable of operating on anonymized identifiers. Finally, due to the volume reduction performed by NetFlow, our approach is very light-weight and capable of operating at very high traffic rates. We refer to our approach as being *remote*, based on the locality of the observation point: we expect the NAT detector to be outside the NAT network. No such passive remote NetFlow based approach exists today.

The remainder of the paper is structured as follows: Section 2 introduces the necessary background on NAT, NetFlow and machine learning algorithms. Section 3 discusses related work in the field of NAT detection. Section 4 sketches our novel approach. Section 5 describes our testing environment and discusses the performance of our approach. Finally, the conclusion and outlook for future work is given in Sect. 6.

2 Background

This section introduces fundamentals of NAT (Sect. 2.1), NetFlow (Sect. 2.2) and machine learning (Sect. 2.3) which will be required in the rest of the paper.

2.1 Network Address Translation

Network Address Translation (NAT) [1] is a technique commonly employed in computer networks to hide a number n of computing resources behind a number $m \leq n$ of IP address resources. For NAT to work, at least one NAT gateway with two network interfaces, i.e. internal and external network interface, is required. By performing NAT, the NAT gateway connects the inside to the outside network and vice versa. The NAT gateway's external interface is configured with m IP

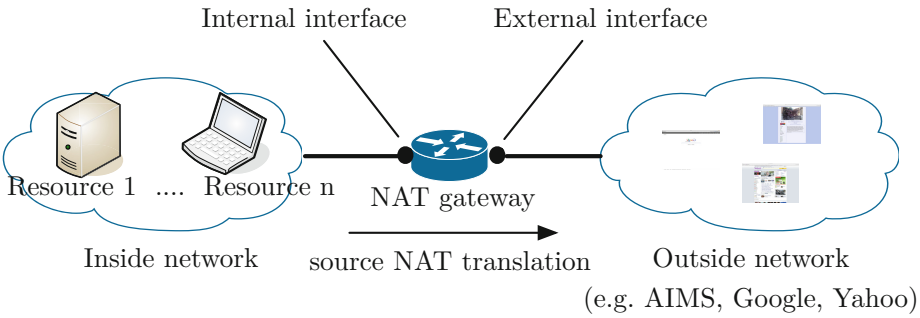

Fig. 1. A typical source NAT setup

addresses routable in the outside network. Inside, $n \geq m$ IP addresses may be used that potentially cannot be routed in the outside network. To connect the inside network with the outside network, the NAT gateway has to rewrite source or destination IP address of any outgoing or incoming IP packet, respectively. This process is commonly referred to as *address translation*. In order to avoid collisions and to correlate sessions when addresses are translated, NAT gateways maintain a *session table* containing a mapping of the original IP address, the NAT IP address it got rewritten to and the rewrite direction, i.e. inside vs. outside. We differentiate between the following NAT mechanisms:

Source NAT. With source NAT, the IP addresses of the inside network are hidden from the outside network. For every outgoing packet, the NAT gateway has to rewrite the source IP address to one of the addresses assigned to its external interface. Source NAT is typically employed to connect one or more computing resources to the Internet (e.g. DSL dial-up). A typical source NAT setup is depicted in Fig. 1.

Destination NAT. In case of destination NAT, the NAT gateway translates destination IP addresses of incoming IP packets to specific addresses of the inside network. This mode of operation is commonly employed to forward specific ports to a specific host serving incoming requests on these ports.

Static NAT. Static NAT is a basic mode of operation in which the NAT gateway always translates one original IP address to the same NAT IP address. Hence, the NAT gateway maintains a static bijection.

Dynamic NAT. Dynamic NAT is a more complex mode of operation in which the NAT gateway maps IP addresses dynamically to a pool of NAT IP addresses. The condition for a mapping to happen is that the NAT IP address is currently not bound to a different original IP address, i.e. that there is no active entry in the session table for a specific NAT IP address. The actual mapping process is vendor specific.

NAT Overload. NAT overload is commonly also referred to as Port and Address Translation (PAT) and is typically employed in source NAT scenarios. In contrast to static and dynamic NAT, a NAT gateway configured with

NAT overload not only translates the original IP address to the NAT IP address, but also rewrites source or destination port numbers of the underlying flows. This mode of operation is typically deployed in cases where $n \gg m$. Additionally, this mode of operation is commonly found on small-office-home-office (SOHO) NAT gateways.

For the remainder of this work, when referring to NAT, we refer to source NAT with NAT overload configured in a usual way.

2.2 NetFlow

NetFlow is a technology first introduced by Cisco Systems [2] that enables monitoring of network flows. A network flow is a unidirectional data stream between two communicating hosts that shares common attributes at the network (L3) and transport layers (L4). A network flow comprises packets offering the same source and destination IP addresses, source and destination port numbers, and the layer 4 protocol type number within a given period of time. More formally, we write a network flow as a 5-tuple, which we denote by f, i.e. we have $f = (srcIP, dstIP, srcPort, dstPort, L4Proto)$. The attributes constituting a network flow f are called *flow keys* [3]. Using NetFlow, it is possible to collect and export statistics corresponding to network flows on IP routers. Specifically, using currently widely deployed versions of NetFlow, i.e. NetFlow versions 5 and 9, routers can export information on bytes and packets transferred, TCP flags set as well as start and end time of a flow and its time duration.

Being a compression function, NetFlow greatly reduces the amount of data to process. This is due to the fact that NetFlow not only aggregates consecutive IP packets sharing the same flow keys within a specific period of time, but also - in contrast to e.g. the Deep Packet Inspection (DPI) based approaches - does not export any payload information. As benefit of this property, using NetFlow for NAT detection instead of full packet captures protects privacy of communicating hosts better than DPI based approaches can do. However, this limited amount of per-packet information possibly makes the detection process more difficult.

2.3 Machine Learning

As already mentioned in Sect. 1 our aim is to apply two machine learning algorithms to behavior statistics derived from NetFlow. The following two paragraphs briefly describe foundations of Support Vector Machines (SVMs) and the C4.5 decision tree algorithm. Finally, we formally describe the measures we use to compare our results.

Support Vector Machines. An SVM is a supervised learning approach that can be applied to linear and non-linear classification problems [4]. These are suitable for learning on a large set of examples and because of that they are commonly used in traffic classification [5,6,7] and anomaly detection systems [8,9,10]. Unlike other classification algorithms, SVMs tend to use all the available features by combining them in a linear way. In general, an SVM tries to fit

a model to a given classification problem by computing a set of maximum-margin hyperplanes separating the classes of interest. If the classes of a specific classification problem cannot be separated linearly using the feature vectors at hand, SVMs transform the input data, i.e. feature vectors, to higher-dimensional feature vectors by applying a kernel function $K(x, y)$ to it. This process is referred to as the kernel trick.

C4.5 Decision Tree. The C4.5 algorithm [11] is one of the most popular decision tree algorithms. A decision tree is a tree structure that consists of decision nodes and leaves. Decision nodes consist of tests on features and leaves represent classes. C4.5 tries to fit a model to a given classification problem by recursively generating such trees. For each decision node, C4.5 computes normalized information gain of the remaining features. A feature is chosen as a decision criterion, i.e. as a new child node, if normalized information gain of a test computed on that feature is the maximum. This process is repeated for each decision node.

Performance Measures. In machine learning, performance of algorithms is usually expressed in the number of *false-positives*, *true-positives*, *false-negatives* and *true-negatives*. Let $\mathcal{X} = \{(\mathbf{x_1}, e_1, c_1), (\mathbf{x_2}, e_2, c_2), \ldots, (\mathbf{x_n}, e_n, c_n)\}$ denote a data set of n k-dimensional feature vectors $\mathbf{x_i} \in \mathbb{R}^k$ $(i, k, n \in \mathbb{N})$ with corresponding a-priori labels $e_i \in \{0, 1\}$ and labels $c_i \in \{0, 1\}$ assigned a-posteriori by a specific binary classifier $\mathcal{C} : \mathbb{R}^k \to \{0, 1\}$. Then, we measure false-positives $FP = \{\mathbf{x_i} \in \mathcal{X} | e_i = 0 \wedge c_i = 1\}$, true-positives $TP = \{\mathbf{x_i} \in \mathcal{X} | e_i = 1 \wedge c_i = 1\}$, false-negatives $FN = \{\mathbf{x_i} \in \mathcal{X} | e_i = 1 \wedge c_i = 0\}$ and true-negatives $TN = \{\mathbf{x_i} \in \mathcal{X} | e_i = 0 \wedge c_i = 0\}$. Based on these definitions, we compute $accuracy = \frac{|TP|+|TN|}{|TP|+|TN|+|FP|+|FN|}$ of our classifier $\mathcal{C}$, where $|.|$ denotes the cardinality of a finite set. For the remainder of this work, we use accuracy as a measure to compare results obtained from different classifiers.

3 Related Work

NAT detection is a long lasting field of research. As such, different active NAT detection approaches, i.e. approaches actively sending and receiving IP packets, were proposed in the past [12,13]. These approaches perform NAT detection from the inside network, usually initiated by a specific application. In contrast to this, our goal is a passive NAT detection from the outside network.

Several approaches for passive NAT detection from the outside of a network already exist. However, most of them rely on raw packet information. For example, Bellovin [14] presents, a method relying on special packet header fields, like for example the IP ID or TCP window size. Moreover in [15] a method for detecting NAT based on instant messaging traffic of different active hosts is described. In addition, Kohno et al. [16] present a method to detect NAT based on passively fingerprinting devices by detecting clock skews in physical devices. Furthermore, recent research regarding tethering detection in mobile broadband networks contributes to the NAT detection problem as well. Schulz et al. [17] describe a tethering detection approach combining and applying different techniques on three different classes of data, namely network layer data,

application layer data as well as behavior and meta data. Even though their approach aims to perform the detection on the ISP side, it is rather specialized to mobile networks.

In contrast to all previous mentioned methods, our approach works completely passively and is based on NetFlow data only. In Krmíček et al. [18], a NetFlow based system for NAT detection is introduced. However, [18] utilizes IP ID and IP TTL information, which are limited to specific NetFlow versions not commonly available in ISP networks [19]. Rui et al. [20,21] propose approaches to NAT detection and size estimation of the inside network using SVM and directed acyclic graph SVM [22], respectively. Their work relies on 8 features consisting of statistics of transferred IP packets as well as of a subset of flags of the TCP header. As this feature set can be derived from NetFlow data, we consider that work equivalent to our approach in terms of objectives. The set of network traces captured and used by Rui et al. [20] for training and evaluation consists of $1,637,550$ packets of 5 hosts. One of the hosts under observation was not placed behind a NAT gateway ($436,320$ packets), while the remaining four hosts generated NAT traffic ($1,201,230$ packets). Thus, approximately 75% of the traffic used for SVM training and testing was NAT traffic. By applying SVM with a radial basis function (RBF) kernel, Rui et al. achieved a detection accuracy of 71.08%. In order to increase accuracy, Rui et al. removed low-volume entries from their data set and achieved a maximum detection accuracy of 83.21%.

4 Passive Remote Source NAT Detection

The NetFlow based approach we propose in this paper is based on the assumption that traffic generated by multiple users/devices shows different behavior than traffic generated by a single user/device. Thus, we try to model user behavior by compiling vectors v_i of 9 features derived from NetFlow records. The features we use can be categorized as being related to packet information and usage intensity (cf. Tab. 1) and are described in sections 4.2 and 4.3, respectively. None of the features relies on payload information and consequently privacy of monitored users is respected. All features can directly be derived from NetFlow.

4.1 Approach

The approach we propose is essentially based on two steps: first, we build a model of user behavior by training machine learning algorithms on a training data set derived from NetFlow. Afterwards, we apply this model to unknown traffic in order to solve our binary NAT/no-NAT classification problem. In order to be able to apply machine learning algorithms to our problem, we first need to derive descriptive features from the NetFlow records at hand.

Feature Extraction. We compute feature vectors $v_i, i \in \mathbb{N}$ for all NetFlow records f_i within a distinct, non-overlapping time window W_t of length 120 seconds[1]. More specifically, we correlate all flows within the window W_t based

[1] Empirical analysis showed that using a window of length 120 seconds yields the best results.

Table 1. Features used for behavior-based NAT detection

Name	Description	Packet	Intensity
tcp	Number of TCP NetFlow records, if any	x	-
udp	Number of UDP NetFlow records, if any	x	-
dns	Number of DNS NetFlow records, if any	x	-
smt	Number of SMTP NetFlow records, if any	x	-
mai	Number of NetFlow records belonging to email protocols	x	-
syn	Number of NetFlow records with SYN flag set	x	-
rst	Number of NetFlow records with RST flag set	x	-
byt	Bytes transferred in flows	-	x
pkt	Packets transmitted in flows	-	x

on the source IP addresses. For each unique source IP address being present in NetFlow records with start time stamp in the window W_t, we compute features as listed in sections 4.2 and 4.3. Basically, our feature extraction process aggregates multiple distinct NetFlow records and correlates information based on IP addresses. The result is a set F_{W_t} containing one feature vector v_i per distinct IP address.

Model Building. Based on the feature vectors extracted as described above, we compute a model that we later use for solving our binary NAT/no-NAT classification problem. The model is built by feeding machine learning algorithms with labeled feature vectors during a training phase. The labeling is performed a-priori according to expert knowledge (cf. Sect. 5.1). After the model is built, previously unknown feature vectors can be classified.

Classification. For classification, we feed a set of unlabeled feature vectors F_{W_t} induced by the window W_t to a previously trained model. On output, we receive a binary decision indicating NAT or no-NAT being present for a single feature vector, corresponding to a distinct source IP address of the given NetFlow records. This process is continuously repeated over time on a series of consecutive sets of unlabeled feature vectors $\{\ldots, F_{W_{(t-1)}}, F_{W_t}, F_{W_{(t+1)}}, \ldots\}$. As our approach does not store or require any inter-window information, it is robust against changes in the network (e.g. joining or leaving subscribers).

4.2 Packet Features

Packet features are features derived from NetFlow that are directly available in layer 3/4 headers of IP packets. Thus, these features by induction are related to user behavior. The packet features we use are:

tcp. Number of TCP NetFlow records.
udp. Number of UDP NetFlow records.
dns. Number of NetFlow records belonging to DNS, i.e. destination port 53.
smt. Number of NetFlow records belonging to SMTP, i.e. destination port 25.
mai. Number of NetFlow records belonging to Email traffic, i.e. destination port 25, 110, 143 , 587, 993 or 995.

syn. Number of NetFlow records with SYN flag set.
rst. Number of NetFlow records with RST flag set.

We expect these features to be highly dependent on user behavior as these features reflect the type of services a specific user consumes. For example, one user may regularly fetch its emails via IMAP while another user may use POP3 in order to access them. Hence, we assume these features to be able to distinguish NAT and no-NAT setups.

4.3 Intensity Features

The intensity features are very high-level ones and can be directly retrieved from NetFlow records:

byt. Number of bytes exchanged within a flow.
pkt. Number of packets transmitted within a specific flow.

We expect both of these features to highly depend on user behavior and the consumed services as well as on the number of users typically being online at a specific point in time. Specifically, we expect more users to transfer more packets and bytes than a single user and vice versa. Thus, for a single IP address under observation, we expect these features to contribute to solving our binary NAT/no-NAT classification problem.

5 Validation

This section describes the validation process of our approach. First, we describe the setup and the data set used in Sect. 5.1. Afterwards, we describe and discuss the results we achieve in Sect. 5.2.

5.1 Setup

Data Collection. The goal of our experiments was to achieve strong results that indicate performance of our NAT detection approach in real-world environments. Hence, we trained our classifiers on NetFlow data we got from a German ISP. The data were collected in the ISP's network during the time span of 8 days from September 4th, 2012 to September 11th, 2012. All NetFlow records belong to DSL subscriber traffic, with a majority of business customers, and have been anonymized using Xu et. al's cryptography-based prefix-preserving anonymization [23] prior to hand-over in order to further respect privacy of the ISP's subscribers. Our approach does not utilize IP address features for NAT detection. It only uses IP addresses as unique identifiers to correlate related flows within a specific period of time. Hence, our approach is specifically capable of operating completely on anonymized traces in order to protect privacy of end-users much better than, for instance, DPI-based approaches. In total, $6,631,383$

anonymized NetFlow records containing both, NAT and no-NAT traffic, were used for training and validation[2].

Labeling. NetFlow records have been labeled according to expert knowledge of the sponsoring ISP. As the ISP provides managed services, it was able to label the NetFlow records based on the IP addresses of their customers. NetFlow records belonging to no-NAT traffic were labeled *ntf*, while NetFlow records stemming from NAT traffic were assigned a *tf* label. We are aware that this labeling approach may be error prone in a sense that a specific system in a customer's network may exhibit NAT-equivalent behavior (e.g. virtualization). However, the co-operating ISP assured that such systems are not prevalent, if present at all.

Datasets. From this $6,631,383$ NetFlow records, we used the first two days full of NetFlow records, i.e. $1,795,964$, for training and the NetFlow records of the remaining six days, i.e. $4,835,419$, for testing of machine learning algorithms. Derived from this number of NetFlow records we obtained $484,499$ feature vectors in total of which $79,047$ were used for training and $405,452$ for testing of algorithms. For the remainder of this paper, we refer to this data set as DS_1. When analyzing the data at hand, we found that 76.3% of the training NetFlow records belong to no-NAT traffic. However, this majority of NetFlow records resulted in only 17.2% of feature vectors. Similarly, 42% of testing NetFlow records belonging to class no-NAT resulted in 8.3% of feature vectors. The reason for this imbalance is that feature vectors are extracted only if traffic related to a certain IP address occurred. An analysis of the data showed that many single users, i.e. no-NAT traffic, are active only for a few hours across a day. Because of that, no feature vectors could be extracted during several periods of time for no-NAT traffic. In contrast, NAT traffic showed almost constant levels of activity. As of the time of writing, we are not completely sure if this observation is immanent to NAT/no-NAT scenarios and could be exploited as additional feature or is related to the data at hand. Thus, in order to not introduce bias when applying machine learning algorithms to the given data, we derive a balanced reference data set for algorithm training by randomly sampling the feature vectors of the prevalent class NAT. This balanced data set is further referred to as DS_2 and its parameters in comparison to DS_1 are given in Tab. 2. Hence, accuracy achieved with this balanced data set DS_2 has to be regarded as lower bound.

5.2 Results and Discussion

We apply two well-known machine learning algorithms, i.e. Support Vector Machines and C4.5 decision tree algorithm[3], to the labeled data sets DS_1 and DS_2 described in the previous section. As sketched in Sect. 4.1 we first train models

[2] We are permitted to share the data set used for validation with researchers in anonymized and partially sanitized format under NDA. Please contact the first author if you would like to have access to this data set for related research.

[3] We use the WEKA J48 implementation of C4.5 and WEKA's SVM implementation.

Table 2. Data sets used and accuracy of behavior based approach

	NetFlow records			Feature vectors			Accuracy	
Data set	Total	Training	Testing	Total	Training	Testing	C4.5	SVM
DS_1	6,631,383	1,795,964	4,835,419	484,499	79,047	405,452	95.35%	95.10%
DS_2	6,631,383	1,795,964	4,835,419	90,864	25,256	65,608	89.35%	81.29%

using our training data sets. Afterwards we apply the testing data sets to models derived during training in order to validate the results. The classification results are depicted in Tab. 2.

When using the imbalanced data set DS_1 we achieve an accuracy of 95.35% in case of C4.5 and an accuracy of 95.10% for SVM. When applying C4.5 and SVM to the balanced data set DS_2 we achieve an accuracy of 89.35% and 81.29%, respectively. The results show that learning with biased data indeed leads to better results. As of the time of writing, however, it is unclear whether imbalance found in DS_1 should indeed be considered bias or is a valuable feature of NAT that could be exploited for NAT detection.

When comparing C4.5 and SVM, the results indicate that C4.5 seems to be more robust to changes in data than SVM as accuracy of C4.5 is only 6 percentage points lower, while it decreases by 13.81 percentage points in case of SVM. Thus, C4.5 in this experiment seems to generalize much better than SVM does. Additionally, time to build the model, i.e. time of training, of C4.5 is much lower (avg. 7.32 seconds) than model build time of SVM (avg. 1366.99 seconds). Similarly, classification with C4.5 requires only 2.22 seconds on average while SVM requires 1371.46 seconds on average for all given feature vectors. These results show that C4.5 can indeed be used to solve our binary NAT/no-NAT decision problem in real-time. Using C4.5, we achieve a lower-bound of accuracy of 89.35% which outperforms the existing approach of Rui et al. [20].

6 Conclusion and Future Work

NAT detection is important to enforce management and security policies in today's networks. In this paper, we present a novel remote source NAT detection approach based on user behavior statistics derived from NetFlow data, working completely passively. In order to solve our binary NAT/no-NAT classification problem, we train machine learning algorithms using 9 distinct features related to user behavior. All features can directly be derived from NetFlow records and do not depend on any payload information. As our approach does not require IP addresses to be valid, IP address anonymization can be applied to further increase privacy of end users. Validation of our approach is performed on real NetFlow traces sponsored by a German ISP and shows that our approach is capable of detecting NAT with a lower-bound accuracy of 89.35% in not more than 120 seconds, after initial training.

During our work, we recognized two additional phenomena of NAT that we will exploit in future work in order to develop a combined NAT detection

approach with increased accuracy: First, we noticed that consumer NAT gateways typically perform source port translation on outgoing packets, i.e. apply NAT overload. Translation is performed in a deterministic way according to specific programmatic conditions. We will investigate if such regularity can be spotted and correlated to NAT. Second, we observed that SYN packet size can be extracted from NetFlow records in case of unsuccessful communication attempts (e.g. connection trials to unavailable email or web servers). As different operating systems show differently sized SYN packets, we plan on using this observation to infer the number of different operating systems emitting traffic for one and the same IP address. We believe this to be a strong indicator for NAT. Furthermore, we will be deploying our approach in the data sponsoring ISP's network for further evaluation and improvement. We will also be working towards estimating the size of an inside NAT network in order to improve, for instance, botnet size estimation.

Acknowledgment. This work has been supported by the German Federal Ministry of Education and Research under grant number 03FH005PB2 (INSAIN) and by CASED.

References

1. Egevang, K., Francis, P.: The IP Network Address Translator (NAT). Request For Comments 1631, Informational (1994)
2. Cisco Systems Inc.: NetFlow Services Solutions Guide. Internet resource, `http://www.cisco.com/en/US/docs/ios/solutions_docs/netflow/nfwhite.html`
3. Claise, B., Bryant, S., Leinen, S., Dietz, T., Trammell, B.H.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. Request For Comments 5101 (Proposed Standard) (2007)
4. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT 1992, pp. 144–152. ACM, New York (1992)
5. Dyer, K., Coull, S., Ristenpart, T., Shrimpton, T.: Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In: 2012 IEEE Symposium on Security and Privacy (SP), pp. 332–346 (2012)
6. Li, P., Wang, Y., Tao, X.: A Semi-Supervised Network Traffic Classification Method Based on Incremental Learning. In: Lu, W., Cai, G., Liu, W., Xing, W. (eds.) Proceedings of the 2012 International Conference on Information Technology and Software Engineering. LNEE, vol. 211, pp. 955–964. Springer, Heidelberg (2013)
7. Tabatabaei, T., Karray, F., Kamel, M.: Early internet traffic recognition based on machine learning methods. In: 2012 IEEE Canadian Conference on Electrical Computer Engineering (CCECE), pp. 1–5 (2012)
8. Francois, J., Wagner, C., State, R., Engel, T.: SAFEM: Scalable analysis of flows with entropic measures and SVM. In: 2012 IEEE Network Operations and Management Symposium (NOMS), pp. 510–513 (2012)
9. Hsu, C.H., Huang, C.Y., Chen, K.T.: Fast-Flux Bot Detection in Real Time. In: Jha, S., Sommer, R., Kreibich, C. (eds.) RAID 2010. LNCS, vol. 6307, pp. 464–483. Springer, Heidelberg (2010)

10. Barthakur, P., Dahal, M., Ghose, M.: A Framework for P2P Botnet Detection Using SVM. In: 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp. 195–200 (2012)
11. Quinlan, J.R.: C4.5: programs for machine learning, vol. 1. Morgan Kaufmann (1993)
12. Rosenberg, J., Manhy, R., Matthews, P., Wing, D.: Session Traversal Utilities for NAT (STUN). Request For Comments 5389 (Proposed Standard) (2008)
13. Wei, Y., Yamada, D., Yoshida, S., Goto, S.: A New Method for Symmetric NAT Traversal in UDP and TCP. Network 4, 8 (2008)
14. Bellovin, S.M.: A technique for counting natted hosts. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement, IMW 2002, pp. 267–272. ACM, New York (2002)
15. Bi, J., Zhang, M., Zhao, L.: Security enhancement by detecting network address translation based on instant messaging. In: Zhou, X., Sokolsky, O., Yan, L., Jung, E.-S., Shao, Z., Mu, Y., Lee, D.C., Kim, D.Y., Jeong, Y.-S., Xu, C.-Z. (eds.) EUC Workshops 2006. LNCS, vol. 4097, pp. 962–971. Springer, Heidelberg (2006)
16. Kohno, T., Broido, A., Claffy, K.C.: Remote Physical Device Fingerprinting. IEEE Transactions on Dependable Secure Computing 2(2), 93–108 (2005)
17. Schulz, S., Sadeghi, A.R., Zhdanova, M., Mustafa, H., Xu, W., Varadharajan, V.: Tetherway: a framework for tethering camouflage. In: Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC 2012, pp. 149–160. ACM, New York (2012)
18. Krmíček, V., Vykopal, J., Krejčí, R.: Netflow Based System for NAT Detection. In: Co-Next Student Workshop 2009: Proceedings of the 5th International Student Workshop on Emerging Networking Experiments and Technologies, pp. 23–24 (2009)
19. Steinberger, J., Schehlmann, L., Abt, S., Baier, H.: Anomaly detection and mitigation at Internet scale: A survey. In: Proceedings of the 7th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2013). Springer (2012)
20. Rui, L., Hongliang, Z., Yang, X., Yixian, Y., Cong, W.: Remote NAT Detect Algorithm Based on Support Vector Machine. In: International Conference on Information Engineering and Computer Science, ICIECS 2009, pp. 1–4 (2009)
21. Rui, L., Hongliang, Z., Yang, X., Shoushan, L., Yixian, Y., Cong, W.: Passive NATted Hosts Detect Algorithm Based on Directed Acyclic Graph Support Vector Machine. In: International Conference on Multimedia Information Networking and Security, MINES 2009, vol. 2, pp. 474–477 (2009)
22. Platt, J.C., Cristianini, N., Shawe-taylor, J.: Large Margin DAGs for Multiclass Classification. In: Advances in Neural Information Processing Systems, pp. 547–553. MIT Press (2000)
23. Xu, J., Fan, J., Ammar, M., Moon, S.B.: On the design and performance of prefix-preserving IP traffic trace anonymization. In: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, IMW 2001, pp. 263–266. ACM, New York (2001)