2014 IEEE Security and Privacy Workshops

Can we identify NAT behavior by analyzing Traffic Flows?

YaseminGokcen
Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada
gokcen@cs.dal.ca

VahidAghaeiForoushani
Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada
vahid@cs.dal.ca

A. NurZincir-Heywood
Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada
zincir@cs.dal.ca

Abstract—**It is shown in the literature that network address translation devices have become a convenient way to hide the source of malicious behaviors. In this research, we explore how far we can push a machine learning (ML) approach to identify such behaviors using only network flows. We evaluate our proposed approach on different traffic data sets against passive fingerprinting approaches and show that the performance of a machine learning approach is very promising evenwithout using any payload (application layer) information.**

***Keywords*—Network address translation classification, traffic flows, traffic analysis, machine learning.**

I. INTRODUCTION

The usage of Network Address Translation (NAT)devices is very common among the devices such as computers, laptops and smart phones connecting to the Internet. While NAT devices are generally used in local area networks (LAN), they can also be used just for one computer. In home networks, most Internet Service Providers (ISPs) give WiFi-enabled NAT home gateways to their users. Thus, when users can connect their devices to the Internet, theirprivate Internet Protocol (IP) addresses are hidden on the Internet since they are encapsulated with a public IP address that the NAT provides.

Basically, NAT allows a single device, such as a proxy, to act as an agent between the Internet and a private network. This means that only a single IP address is required to represent one or more computers to the rest of the world. Thus, NATs are used for many reasons such as a solution to the shortage of IPv4 addresses or for security and privacy reasons. The latter is the focus of this work.In this case, to be represented by one public IP addressgives anonymity within the group of computers behind a NAT device. This could be advantageous tosome users, since their private IP addresses cannot be identified at a glance.All these advantages enable the NAT usage to be common for network management purposes. However, the same reasons also make itattractive and useful for malicious users who want to hide their real identities. Hence, NAT usage increases both in legitimate and in malicious environments.

Moheeb et al. [18]showed that the initial exploit in multi-stage infections was likely some form of a shellcode containing a Uniform Resource Locator (URL) that hosts the malware binary. In their research, such a URL pointed back to the source that sent the exploit in the first stage. Using this, they could determine which sources were located behind NAT devices by parsing the log of collected URLs. They could then extractthose sources that use local IP addresses in the URL sent to the victim.They showed that the widespread use of NAT had significant implications on how different families of malware spread on the Internet.

Thus,identifying the NAT devices becomes more and more important to understand and to detect the malicious behavior in traffic and application usage. In this research, our goal is to explore whether we can find specific patterns in the network traffic that will enable us to identify NAT like behaviors. To this end, we propose a ML based approach to automatically find patterns indicating NAT usage without using IP addresses, port numbers or payload information. In doing so, our aim is to not only analyze how far we can push a ML approach but also to achieve a well generalized classifier, which can work under encrypted payload and traffic filtering (based on IP address or port number) conditions.We evaluate our proposed system on two different data sets and compareit to other approaches used in the field. Our results demonstrate that the proposed system is very promising in identifying NAT device behaviors by analyzing traffic flows. This enables us to analyze both encrypted and non-encrypted traffic since traffic flows are only based on the packet header features.The remainder of this paper is organized as the following. Section II reviews the literature on NAT related research. Section III describes our proposed system and the methodology we followfor our evaluations. Section IV presents the empirical results and our observations. Finally, Section V draws conclusions and discusses the future work.

II. LITERATURE REVIEW

The identification of NAT devices and the number of end users behind such devices is a relatively new research area. To this end, different algorithms were proposed, but generally, researchers used some form of passive Operating System (OS) fingerprinting to identify NAT behaviors. To achieve this,they analyzecertain parameters within the TCP/IP (Transmission

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

DOI 10.1109/SPW.2014.28

Control Protocol/ Internet Protocol) protocol stack and most of the time conduct experiments with simulated NAT data sets.

Examples of passive methods includeBellovinet al.[8] who identified that consecutive packets carry sequential IP Identification (ID) fields, which were included in the IP header and generally were used as counters. Therefore, theyconcluded that it was possible to count the string values of those IP ID fields to find the number of hosts. However, such an approach might have some complications when faced with packets with zero IP IDs or packets using byte-swapped counters. Moreover, the recent versions of OpenBSD and FreeBSD use pseudo-random number generator for the IP ID fields. So counting such values will not work accurately on these OSs.Beverlyet al.[7] proposed a classifierto infer the OS passively and find the number of hosts behind a NAT. They used Time-To-Live (TTL), Do-not-fragment (DF), Window-size and SYN-size features. Miller et al.[6] analyzed thetcpdump packets. They checked certain fields, namely Type of Service (TOS), total length, IP ID, TTL, source port, window and TCP options in the TCP/IP header to fingerprint different OSs. The idea is that if an IP address has more than one OS associated with it, this might indicate a NAT device with different OSs behind it. Indeed, it is easy to see that such an approach will give false alarms if a computer has more than one OS on it.

Moreover, more active methods are also proposed. Murakami et al. [3] focused on the Medium Access Control (MAC) address of a device and proposed a NAT router, which relays the MAC address of the computers based on FreeBSD.So ifthe proposed NAT relaying machine is used on a network,then it is straight forward to identify the NAT device.Ishikawa et al. [1] proposed to identify the computers behind a NAT router with proxy authentication on a proxy server. Their target application was WWW. They used realm field in the authentication header by associating with a MAC address of a client computer after authentication is succeeded. That realm is shown to the user as a prompt message. Their proposed system requires Java Runtime Environment (JRE) on each client machine. They assumed that a web browser always adds the authentication header to its request message when authentication has succeeded. Even though this method is successful, it could only be used for their target applicationand the proxy conditions require the JRE code the authors developed.

On the other hand, Rui et al. [4] proposed the use ofa Support Vector Machine (SVM) based classifier to detect the presence of a NAT in agiven traffic trace. They captured traffic on five different hosts that were potential NAT devices. Then they trained a SVM classifier on the captured traffic to be able to detect the NAT device among the five potential hosts.They collected different statistics on a host to show its activeness. Their hypothesis was that the hosts behind a NAT device send more bytes, need more network connections, visit more web sites,and produce more complex traffic traces compared to an ordinary host. Therefore, a host behind a NAT device would have higher activity value than an ordinary one.They labeled their traffic data as ordinary hosts and hosts behind a NAT. Then, they applied binary classification using the SVM.Even though this is an interesting approach to collect traffic on the host that is suspected to be a NAT device. However, collecting such data may not be feasible in practice unless we have full control on such a host.

Maier et al. [2] focused on detecting DSL lines that use NAT devices to connect to the Internet. Their approach is similar to passive fingerprinting and is based on the IP TTL and the HTTP user agent strings. They extracted the operating system and the browser family and its versions from the HTTP user agent strings. Indeed, this necessitates deep packet inspection into the payload of a packet. However, if they do not have access to the payload information, then they only use the IP TTL information. They analyzed the user agent strings of typical browsers and they ignored the ones, which came from mobile devices and gaming consoles. They found 10% of DSL lines had more than one user active at the same time, and that 20% of the lines have multiple hosts that were active within one hour of each other. These results indicated the potential NAT devices in their DSL data sets.

In summary, Maier et al. [2]'s approach, seems to be the best performingapproach reported in the literature albeit its requirement for payload information. Thus, in this paper, we re-engineered their approach to understand its advantages and disadvantages. In addition, we employed their system as the representation of the state-of-the-art in our evaluations against our proposed system. Finally, our aim is to study whether we can learn general enough patterns to identify the behavior of NAT devices among the traffic received by the computer that is under analysis. This makes it very different from Rui et al's work [4], which analyzes the traffic of a host to determine whether it is a NAT device or not.To the best of our knowledge, our work is the first one aiming to evaluate these different approaches on different traffic traces from different networks to identify the potential NAT devices.

III. METHODOLOGY

In this research, we aimed to evaluate two different approaches on identifying potential NAT devices on given traffic traces. These approaches are:our proposedMLbased approach and the passive fingerprintingapproachthat analyzesspecific parameters in a given network traffic trace. For the proposed approach, we employ the C4.5and Naive Bayes learning techniques as our classifiers. The reason we chose these two learning algorithms are two folds: C4.5 learning technique provides the solution it learns from the data in a tree form using if-then-else format. This makes it easy for a human expert to analyze the solution and to understand what the algorithm learned. In other words, the solution is no longer a black box. On the other hand, Naïve Bayes is one of the well known statistical learning algorithms (albeit with an opaque solution) so it naturally represents a standard baseline classifier for this work. As for the passive fingerprinting approach, we re-engineered and employed the algorithm introduced by Maier et al. [2] as the representative state-of-the-art technique.

For bothapproaches, we employ the same data sets including both encrypted and non-encrypted traffic from two different organizations. The following describes the data sets, and the algorithms employed as well as the experiments performed in this work.

A. Data sets Employed

In this research, two different data sets from two different organizations (networks), namely Nims-NAT and Partner-NAT,areemployed to evaluate the aforementioned approaches. These are traffic data sets in the form of *tcpdump*log files without any payload information.Our first data set, Nims-NAT also includes the web access log files, which belong to the same time period. Nims-NAT data is collected over a week on November 2012 on our network. This data set is labeled as (i) NAT flows; and (ii) OTHER flows where we know the ground truth about the NAT devices.

Our second data set is provided to us with the ground truth (in terms of NAT flows vs. OTHER flows) by our industrial partner, which is a medium sized private company. We will refer to this data set as Partner-NAT hereafter. Given the privacy issues related to this data set, we will not be able to provide any further details about the Partner-NAT. However, Table I presents some properties of bothNims-NAT and Partner-NAT data sets.

B. *Features Employed*

In this work, we converted our packet based traffic traces (tcpdump files) totraffic flows. To this end, NetMate[15]open source tool is employed to generate the flows and compute the statistical features for each flow.Once the flows are generated, we do not usethe source and destination IP addresses as well as the source and destination port numbersin our feature set to represent the flow traffic to our classifiers. We think that such information can bias the results. It is well known that port numbers can be assigned dynamically and IP addresses can be anonymized very easily. One can say that in some ways, NATs and proxies are already doing this for free. Moreover, filters can be set to block or choose certain traffic. Our aim here is to find patterns (in other words signatures) automatically without using any biased features. Indeed, to be able to apply our approach both to the encrypted and the non-encrypted traffic, we do not employ any payload (application layer) information as features in our proposed ML based approach. However, such information is employed for the passive OS fingerprinting as Mailer et al.described in [2].In the following, we discuss the features in more detail.

1) *Features for the Proposed Approach - Netmate Features*

NetMate[15] is an open source flow generator. In this case, flows are bidirectional and the first packet of the flow identified byNetmate determines the forward (source to destination) direction. A flow can be uniquelyidentified by five parameters within a certain time period. These parameters are source and destination IP addresses, source and destination port numbers and protocol. Netmate considers only the UDP and the TCP flows. Moreover, the UDP flows are terminated by a flow timeout, whereas the TCP flows are terminated upon proper connection teardown or by a flow timeout, whichever occurs first.The flow timeout value employed in this work is 600 seconds as recommended by the IETF [16]. The Netmate features that we used in our experiments are shown in Table II.

2) *Features for the Passive Fingerprinting Approach*

As discussed earlier, we re-engineered the passive fingerprinting approach of Maier et al. as it is described in [2]. In their passive fingerprinting approach, some features require access to the payload (application level) information whereas others do not. We detail these features below.

a) *PacketHeader Base Features – Time to Live (TTL)*

Networking stacks of OSsuse well-defined initial IP TTL values (ttl_{init}) in outgoing packets. For instance, Windows uses 128, MacOS uses 64 andDebian based systems use 64, too.The TTL field of the IP header is defined to be a timer limiting the lifetime of the IP datagram. It is an 8-bit field and may be implemented as a counter or a timestamp. Each router (or other modules) that handles a packet must decrement the TTLby at least one, even if the elapsed time was much less than a second.When a router forwards a packet, it must reduce the TTL by at least one. Thus, it isassumed that if there is a machine routing in the network, in some ways a NAT device falls under this category, it will decrement the TTL values for each packet that passes through.

TABLE I. Number of flows in the data sets employed in this work

			The Number of Flows		
			NAT	**OTHER**	**TOTAL**
DATASETS	**Nims-NAT**	**Training**	9126	9126	18252
		Testing	3042	156199	159241
		Total	12168	165325	177493
	Partner-NAT	**Training**	9126	9126	18252
		Testing	90116	35348	125464
		Total	99242	44474	143716

b) *Packet Payload Base Features – HTTP User Agent String*

The user agent string identifies the browser that is used to access the web. When a user visits a webpage, his/her browser sends the user-agent string to the web server hosting the site that is visited. This string indicates which browser is in use, its version, and otherdetails about the user's system, such as the OS and its version.

For this part (payload information), we utilized the web access log files to be able to extract this information, Table III.Maier et al. [2] analyzed the user agent strings to obtain the OS and the browser information. Then based on this information, they made adecision regarding the presence of a NAT device in the traffic. They limited their analysis to user agent strings from typical browsers such as Firefox, Internet Explorer, Safari and Opera [2].However, when we applied their approach on our data sets, we did not limit their technique only to the typical browsers. In our data sets, we observed many user agent strings from Android based devices, iPhones and iPads so we included them in our analysis.

C. *Passive Fingerprinting Approach*

We evaluated this approach in four different steps based on the utilization of the features to better understand their effect on a given data set.

1) TTL Range

In the simplest form, Maier et al. infer the presence of a NAT device based on the TTL values of packets sent by users. If the TTL is $ttl_{init} -1$ the sending host is directly connected to the Internet (as the monitoring point is one hop away from the device on which the traffic is monitored / analyzed).If the TTL is $ttl_{init} - 2$ then there is a routing device (i.e., a NAT device) in the users' premises. Hereafter, werefer to this technique as "L1".

Although L1 techniquecan be used to detect the presence of a NAT device for some networks, it also has some limitations. These may prevent the detection of a NAT device under the following conditions:

- L1assumes that the number of hops between the machine on which the traffic is captured and the machine on which the analysis is made, is known. Because, only then the TTL values used can be interpreted accurately to detect a NAT device. Otherwise, L1 technique cannot work accurately.
- L1 assumes that the NAT devices decrement the TTLs for each packet that passes through them. However some NAT implementations might not decrement the TTL values for some reason or another such as hiding the network topology.
- In Maier et al.'s work, the data set employed is from an ISP so the analysis (monitoring) is performed on the residential users' traffic of the ISP. In that case, the traffic coming from residential ISP users naturally goes through a device, which performs the NAT as well as the DNS services. They assume the traffic gets NATed, if they see an IP address that sends DNS packets with a special TTL value. Moreover, they assume that TTL value is only used for DNS packets. However, in some networks, the DNS and the NAT services might not be on the same server.Therefore, in these cases, the TTL values will not be as accurate as the ones seen in Maier et all's data sets.

2) *TTL Range and the Distinct TTL Values Per IP Address*

In this case, not only the TTL range is observed to detect a NAT device but also thenumber of distinct TTL values is observed per IP address.Given that Windows uses a $ttl_{init}$of 128, MacOSX and Linux use 64, and these ranges are far enough apart to distinguish between them, observed TTL values can be used to distinguish between Windows and non-Windows OSs. Thus, if more than one TTL value range is observed for one IP address, then that IP address may belong to a NAT device. Hereafter, we refer to this technique as "L2".

Although utilizing this techniquecan give a better performance than L1for detecting a NAT device, it still has the following limitations:

- Similar to L1, if the users reconfigure their systems to use a different TTL policy, then this system cannot infer the presence of a NAT device based on the different TTL values.
- When there are two completely different operating systems (eg. Linux and Windows) on the same host, this approach would see two different TTL values (e.g. 64 and 128). So it would infer that there is a NAT device in this traffic, even though there is not.

TABLE II.Features Employedby the Proposed Approach

No	Feature Name	Abbreviation
1	The protocol (i.e. TCP=6, UDP=17)	proto
2	Total packets in the forward direction	total_fpackets
3	Total bytes in the forward direction	total_fvolume
4	Total packets in the backward direction	total_bpackets
5	Total bytes in the backward direction	total_bvolume
6	Size of the smallest packets sent in the forward direction	min_fpktl
7	Mean size of packets sent in the forward direction	mean_fpktl
8	Size of the largest packet sent in the forward direction	max_fpktl
9	Standard deviation from the mean of the packets sent in the forward direction	std_fpktl
10	Size of the smallest packet sent in the backward direction	min_bpktl
11	Mean size of the packet sent in the backward direction	mean_bpktl
12	Size of the largest packet sent in the backward direction	max_bpktl
13	Standard deviation from the mean of the packets sent in the backward direction	std_bpktl
14	Min. amount of time between two packets sent in the forward direction	min_fiat
15	Mean amount of time between two packets sent in the forward direction	mean_fiat
16	Max. amount of time between two packets sent in the forward direction	max_fiat
17	Standard deviation from the mean amount of time between two packets sent in the forward direction	std_fiat
18	Min. amount of time between two packets sent in the backward direction	min_biat
19	Mean amount of time between two packets sent in the backward direction	mean_biat
20	Max. amount of time between two packets sent in the backward direction	max_biat
21	Standard deviation from the mean amount of time between two packets sent in the backward direction	std_biat
22	The duration of the flow	duration
23	Min. amount of time that the flow was active before going idle	min_active
24	Mean amount of time that the flow was active before going idle	mean_active
25	Max amount of time that the flow was active before going idle	max_active
26	Standard deviation from the mean amount of time that the flow was active before going idle	std_active
27	Min. time a flow was idle before becoming idle	min_idle
28	Mean time a flow was idle before becoming idle	mean_idle
29	Max. time a flow was idle before becoming idle	max_idle
30	Standard deviation from the mean amount of time a flow was idle before becoming idle	std_idle
31	Average number of packets in a sub flow in the forward direction	sflow_fpackets
32	Average number of bytes in a sub flow in the forward direction	sflow_fbytes
33	Average number of packets in a sub flow in the backward direction	sflow_bpackets
34	Average number of bytes in a sub flow in the backward direction	sflow_bbytes
35	Number of times the PSH flag was set in packets travelling in the forward direction	fpsh_cnt
36	Number of times the PSH flag was set in packets travelling in the backward direction	bpsh_cnt
37	Number of times the URG flag was set in packets travelling in the forward direction	furg_cnt
38	Number of times the URG flag was set in packets travelling in the backward direction	burg_cnt
39	Total bytes used for headers in the forward direction	total_fhlen
40	Total bytes used for headers in the backward direction	total_bhlen

TABLE III. Features (and some example cases) Employed by the Passive Fingerprinting Approach

	From Packet Header		From HTTP User Agent		
Approach	TTL	IP	OS	Browser Family	Browser Version
L1	61	-	-	-	-
L2	61	129.173.13.94	-	-	-
L3	125	129.173.13.94	Windows NT 5.1	-	-
L4	125	129.173.13.94	Windows NT 5.1	Firefox	3.0.3

3) *TTL Range, the Distinct TTL Values Per IP Address, and the Different OS Information in the HTTP User Agent Strings*

Given the above constraints and the false alarms they may cause, Maier et al. extended their technique into the HTTP user agent strings (when the information is available) to observe the OS types and their versions. In this case, they assume thata NAT device is more accuratebased on the OS fingerprint. Hereafter, we refer to this technique as "L3".However, thistechnique also has the following limitations:

- If all the hosts in a NAT network use the same type of OS, this technique cannot detect the NAT device.
- When there are two versions of an OSon the same host (e.g. Windows XP and Windows 7), this techniquewould detect one TTL value but two different OS versions. So it would classify them as two separate hosts and would infer that there isa NAT device, even though there is not.

4) TTL Range, the Distinct TTL Values Per IP Address, the Different OS and the Browser Information in the HTTP User Agent Strings

In this case, in order to have more accurate results,the browser type and version are also extracted from the HTTP user agent string to detect a NAT device. This technique aims to minimize the false alarms that may arise from one host having two different versions of the same OS. The assumption behind this technique is that one host might have two different web browsers, but it cannot have two different versions of the same web browser working simultaneously [2]. Hereafter, we refer to this techniqueas "L4".

Detecting the NAT devices and their traffic based on the web browser information in the HTTP user agent strings still has the following limitations:

- When there are several computers behind a NAT device with the same OS and the same browser (e.g. a network in a university lab where all of the computers have the same OS and the same browser), this technique could not classify such traffic as NAT traffic, because it could not find any evidence for different TTL values, OSs, and browsers.
- When one host uses a specific version of a web browser and later it uses another version of the same browser, L4 technique could detect these as NAT devices even though they are not. This may happen when the user updates his/her web browser.
- There are several examples of HTTP user agent strings where they do not have any information about the OS and the web browser of the client.Under such conditions, L4 techniquecould not work accurately.

D. Proposed ML Based Approach

As discussed above, our proposed system is a ML based approach using network flow based features. To this end, we employ two learning techniques: a decision tree classifier, namely C4.5, and a probabilistic classifier, namely Naive Bayes. The following summarizesthe learningtechniques employed.

1) C4.5

C4.5 is an algorithm that generates a decision tree using information gain. A decision tree is a hierarchical data structure for implementing a divide-and-conquer strategy. C4.5 is an efficient non-parametric technique that can be used for both classification and regression problems. C4.5 constructs decision trees from a set of training data applying the concept of information entropy, Eq. (1) [13].The training data is a set, *S*, such that each input of the set is an instance of already classified samples. Each sample in the set is a vector where each element in the vector represents a feature of the sample. C4.5 can split the data into smaller subsets using the fact that each feature of the data can be used to make a decision (one class versus another class). The feature with the highest information gain is used to make the decision of the split.

$$J_m = -\sum_{j=1}^{n} p_m^i \log_2 p_m^i (1)$$

If the split is not pure, then the instances should be split to decrease impurity. There are multiple possible features on which a split can be done. Indeed, this is locally optimal; hence there is no guarantee of finding the smallest decision tree. In this case, the total impurity after the split can be measured by Eq. (2) [13]. In other words, when a tree is constructed, at each step the split that results in the largest decrease in impurity is chosen. This is the difference between the impurity of data reaching node *m*, Eq. (1), and the total entropy of data reaching its branches after the split, Eq. (2). A more detailed explanation of C4.5 algorithm can be found in [9].

$$\boldsymbol{I}_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{k} p_{mj}^i \log p_{mj}^i (2)$$

2) Naive Bayes

Naive Bayesian is a statistical classifier based on Bayes theorem that gives its conditional probability a given class.A Naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. Depending on the precise nature of the probability model, Naive Bayes classifiers can be trained efficiently in a supervised learning approach [14]. A simple Naive Bayes probabilistic model can be expressed as Eq.(3) in the following:

$$P(C|F_1, F_2, \dots, F_n) = \frac{1}{Z} P(C) \prod_{i=1}^{n} P(F_i|C), \qquad (3)$$

where$P(C|F_1, F_2, \dots, F_n)$ is the probabilistic model over the dependent class variable C with a small number of outcomes or classes, conditional on several feature variables F_1 through F_n; Z is a scaling factor dependent only on $F_1, F_2, \dots, F_n$, i.e., a constant if the value of the feature variables are known.More detailed information on the Naive Bayesian algorithm can be found in [14].

IV. Experiments and Results

In this research we have employed C4.5 and Naïve Bayes learning algorithms via an open source tool called Weka[17].We measure the performance of all the techniques employed using two metrics, namely Detection Rate (DR) and False Positive Rate (FPR). DR reflects the number of NAT traffic flows correctly classified. It is calculated using Eq. (4):

$$DR = TP / (TP+FN) \qquad (4)$$

where False Negative (FN) reflects the number of NAT flows incorrectly classified as OTHER flows, i.e. non-NAT flows. On the other hand, FPR reflects the number of OTHER flows incorrectly classified as NAT flows using Eq. (5):

$$FPR = FP/ (FP+TN) \qquad (5)$$

Naturally, a high DR and a low FPR are the desirable outcomes.

A. Performances of the Passive Fingerprinting Approach

We appliedall the passive fingerprinting classifiers, namely L1, L2, L3 and L4 techniques, to our datasets to identify the presence of NAT behavior. These results are discussed in the following.

1) L1

L1 classification technique aims to detect the presence of NAT behaviorbased only on the TTL values present in the traffic traces. As can be seen in Table IV, in this case, the DR is 0% and FPR is 100% for both of the datasets. The reason is thatL1 requires the prior knowledge about the location of the monitoring point. However, we do not have any prior knowledge about the location of the monitoring point in our data sets.

Fig. 1, Fig. 2 and Fig. 3 show the different TTL value ranges for the Nims-NAT data sets.As can be seen from these figures, they do not fall in the range, ttl_{init}-1 and ttl_{init}-3, as given by Maier et al. [2].

Fig. 1. TTL range for Microsoft Windows versions (MS Windows 95/98/98 SE etc.)

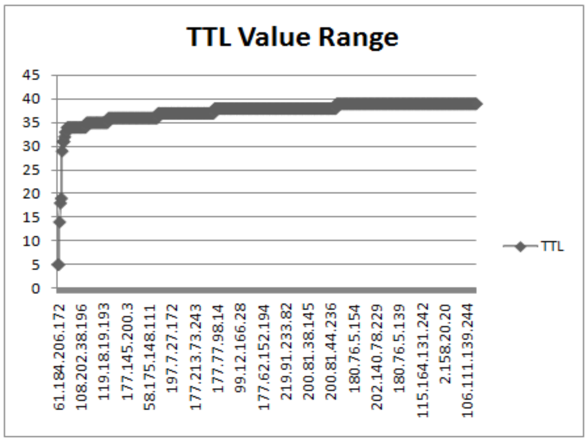

Fig. 2. TTL range for Mac OS X, Unix and Unix like systems

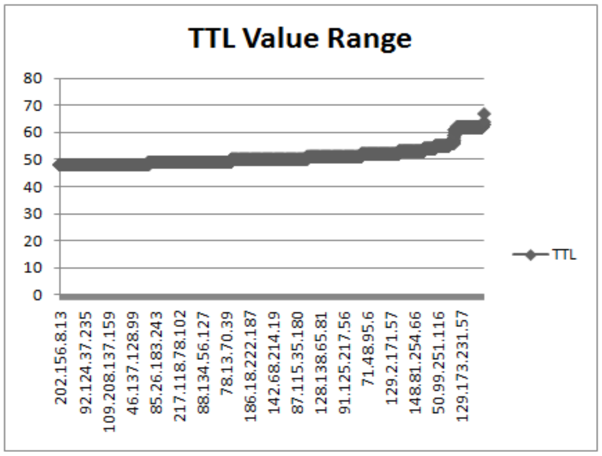

Fig. 3. TTL range for newer Microsoft Windows versions (MS Windows 2000, Vista etc.)

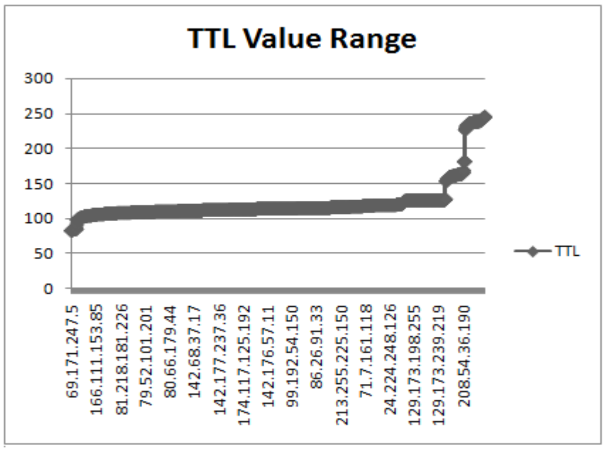

TABLE IV. Test Results on the Nims-NAT and the Partner-NAT data sets by using the L1 classifier

	Class-NAT		Class-OTHER	
Data sets	**DR**	**FPR**	**DR**	**FPR**
Nims-NAT	0%	100%	0%	100%
Partner-NAT	0%	100%	0%	100%

2) L2

L2 classification technique aims to detect the presence of NAT behaviorbased on the TTL range and the distinct TTL values per IP address. Table V presents the DRand FPR results for this classifier on our data sets. In this case, the DR of the NAT behavior in the Nims-NATdata set is 100% because the real NAT devices in this data set has more than one TTL value (64 and 128), so all distinct instances belonging to this IP address are detected. In the Nims-NAT data set, there are 12,168 NAT traffic flows (out of 177,493 flows in total)and all are identified (detected) correctly.However the DR for the Partner-NATdata set is 0%. Since all the OSs that belong to the NAT IP addresses in the Partner-NATdata set have the same OS, there is noNAT device IP address with more than one TTL value in this data set.Moreover, for both theNims-NAT and Partner-NATdata sets, there are some flows that belong to the hosts that have both Windows and Linux OSs. Thus, these hosts IP addresses have more than one distinct TTL valueobserved in the data sets. Thisresults in the L2 technique to identify them as NAT devices and causes theFPR for bothdata sets.

TABLE V.Test Results on the Nims-NAT and the Partner-NATdata sets by using the L2 classifier

	Class-NAT		Class-OTHER	
Data sets	**DR**	**FPR**	**DR**	**FPR**
Nims-NAT	100%	0.16%	99.8%	0%
Partner-NAT	0%	2.7%	97.2%	100%

3) L3

L3 classification technique aims to detect the presence of NAT behavior based on the TTL Range, the distinct TTL values per IP address, and the different OS information in the HTTP user agent strings per IP address. Table VI shows the DR and FPR for this classifier on our data sets. In this case, the FPR of the L3 classifier on the Nims-NAT data set is more than the FPR of the L2 (Table V) classifier on the same data set even though L3 classifier employs payload inspection, i.e. HTTP user agent string. This was not expected so when we analyzed the data set, we found that Nims-NAT data set has some instances that belong to the hosts with two OSs, but both of those OSs are two different versions of the same OS, e.g. Windows XP and Windows 7, so L3 classifierautomatically detects them as NAT devices,which is obviously not correct. As for the Partner-NAT data set, the DR of NAT flows is still 0% because L3 classifier cannot detect the NAT flows coming from hosts that use the same version of the same OS behind the NAT device.

TABLE VI. Test Results on the Nims-NAT and the Partner-NAT data sets by using the L3 classifier

	Class-NAT		Class-OTHER	
Data sets	**DR**	**FPR**	**DR**	**FPR**
Nims-NAT	100%	0.93%	99.6%	0%
Partner-NAT	0	2.7%	97.3%	100%

4) L4

L4 classification technique aims to detect the presence of NAT behavior based on the TTL Range, the distinct TTL values per IP address, the different OS and the browser information in the HTTP user agent strings per IP address.As shown in Table VII, the FPR of L4 classifier on the Nims-NAT data set is very high (6%). The reason is that there is a DHCP server that assigns the IP addresses randomly to the mobile devices (e.g. smartphones and laptops)on this network. These devices have different versions of the same web browser and might end up using the same IP address during different times of the day. In this case, L4 classifier categorizes them as NAT devices even though they are not and hence the high FPR. On the other hand, L4 classifier works much better (DR: 100%, FPR: 3%) on the Partner-NATdata set than the L1, L2 and L3 classifiers.

TABLE VII. Test Results on the Nims-NAT and the Partner-NAT data sets by using the L4 classifier

	Class-NAT		Class-OTHER	
Data sets	**DR**	**FPR**	**DR**	**FPR**
Nims-NAT	100%	6%	93.9%	0%
Partner-NAT	100%	2.7%	97.2%	0%

B. *Performance of the Proposed Approach*

As we have seen in the previous section, each classifier used for passive fingerprinting approach to identify NAT behavior in the monitored data has some drawbacks on one data set or the other.This not only shows that detecting the NAT behavior in the monitored (analyzed) traffic is challenging, but also it shows that there are different NAT behaviors depending on the organization (Nims versus Partner networks). Moreover, each organization's data can be reflected differently depending on at which level it is analyzed i.e. network layer (network traffic flows) versus application layer (HTTP user agent strings). Based on our evaluations presented above, the L2 classifier among the passive fingerprinting techniques was the best for the Nims-NATdata set, while the L4 classifierwas the best for the Partner-NAT data set.

As for the performance of our proposed approach based on the ML classifiers using only network flows,Table II shows the Netmate flow features used to represent the traffic to the learning classifiers. In this case, we trained both the C4.5 and the Naïve Bayes learning algorithms using a portion of the network flow data sets from Nims-NAT and Partner-NAT data sets. To this end, we have employed a balanced training data of network traffic flows from each data set. Weused the Uniform Distribution Filter[1] from Weka to randomly select the training instances to form the training data set for training both of the ML classifiers. Once the classifiers are trained on the trainingdata set,i.e. the sampled data set, we used all the unseen data that is not included in our trainingdata set for testing purposes.

Table Ishows the number of flows used for training and testing for each data set. It should be noted here that these are the same data sets used for the passive fingerprint classifiers. Table VIIIpresents the performance of our proposed approach, where we compared two different ML algorithms for detecting NAT behaviors in Nims-NAT and Partner-NATdata sets. According to these results, the performance results of the C4.5 learning technique seems to be well generalized from one data set to the other. This is achieved without using any IP address, port numbers, TTL data or HTTP user agent strings, i.e. without any application level information.

Table IX shows the most important features that enable the C4.5 based classifier to identify the NAT device behavior at the end of the training phase.

TABLE VIII. Test Results on the Nims-NAT and Partner-NATdata sets by using the proposed approach with flow features

		Class-NAT		Class-OTHER	
		DR	**FPR**	**DR**	**FPR**
Nims-NAT data set	**C4.5 based classifier**	98.7%	3.7%	96.3%	1.3%
	Naive Bayes based classifier	15%	13%	98%	98%
Partner-NAT data set	**C4.5 based classifier**	98%	2.4%	97.6%	2%
	Naive Bayes based classifier	34%	10%	89%	66%

Our proposed system using C4.5 classifier outperforms the L1 classifier of the passive fingerprinting approach on both data sets in terms of DR and FPR. Our proposed system only

[1]Uniform Distribution filter is set in Weka by following this path: weka.filters.supervised.instances.spreadsubsample. This filter produces random subsamples of a data set by using the options options; -S, -M, -W and -X. -M is the maximum class distribution spread. If it is chosen as 1.0, the class values are chosen equally in the manner of uniform distribution.

employs network flow based information. Moreover, it also outperforms the L2 and L3 classifiers on the Partner data set and performs as good as the L4 classifier on both data sets even though L4 classifier employs both the packet header and the HTTP user agent strings information.

As discussed earlier, C4.5 learning technique based classifier has the ability to choose the most appropriate features from a given feature set. This enables us to learn which features of the network flow traffic have contributed to this high performance. Once we analyzed the solution decision tree generated by the C4.5 algorithm, we were able to identify the most helpful features for the classifier to detect different NAT behaviors existing in the network traffic, Table IX.

TABLE IX. The most important Netmate features selected by the proposed system using the C4.5 learning classifier

FEATURES	
Name	**Description**
sflow_bytes	The average number of bytes in a sub flow in the forward direction
total_bvolume	Total bytes in the backward direction
mean_fpktl	The mean size of packets sent in the forward direction
max_active	The maximum amount of time that the flows was active before going idle
min_fpktl	The size of the smallest packet sent in the forward direction
max_bpktl	The size of the biggest packet sent in the backward direction
std_bpktl	The standard deviation from the mean of the packet sent in the backward direction

These features seem to work for both of thedata sets employed in this work. Even though these are the features with the highest weights in the solution, actually our system is based on all 41 Netmate flow features. This also indicates the challenges and different NAT behaviors present in the different data sets.

In summary, these results show that passive fingerprinting classifiers seem to work for certain NAT behaviors better than the others. Moreover, as the NAT behavior gets more unique and challenging, passive approach requires access to the application (payload) information such as HTTP user agent strings to reach a high DR with low FPR. On the other hand, our proposed approach based on the C4.5 decision tree learning classifier, enables us to achieve a high performance (high DR and low FPR) accuracy without using any application level data and generalizing well to different NAT behaviors present in different data sets.

V. CONCLUSION AND FUTURE WORK

In this research, we explored how far we can push a ML based classification approach to identify NAT devices using only network flows. To this end, we represented the traffic as network flows to two ML techniques, namely C4.5 and Naive Bayes, without using IP addresses, port numbers and payload (application) information. We evaluated our approach on two different data sets against four different variants of the passive fingerprinting approach [2], which represents the state-of-the-art techniques. Our results show that the proposed approach using C4.5 learning classifier performs better than the passive fingerprinting approach on both data sets even though the latter uses payload information. This is a very promising result given that payload becomes opaque when encryption is used at the application level. Future work will analyze different NAT behaviors and explore how solutions of the C4.5 based classifier can be converted into automatic signatures.

ACKNOWLEDGMENT

This research is supported by the Canadian Safety and Security Program(CSSP) E-Security grant. The CSSP is led by the Defense Research and Development Canada, Centre for Security Science (CSS) on behalf of the Government of Canada and its partners across all levels of government, response and emergency management organizations, non-governmental agencies, industry and academia.

REFERENCES

[1] Y. Ishikawa; N. Yamai; K. Okayama; M. Nakamura.; , "An Identification Method of PCs behind NAT Router with Proxy Authentication on HTTP Communication," Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on , vol., no., pp.445-450, 18-21 July 2011.

[2] G. Maier, F. Schneider, and A. Feldmann. NAT Usage in Residential Broadband Networks. Proceedings of the 12th International Conference on Passive and Active Network Measurement (PAM 2011), Atlanta, Georgia, March 2011.

[3] R. Murakami; N. Yamai; K. Okayama; , "A MAC-address Relaying NAT Router for PC Identification from Outside of a LAN," Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on , vol., no., pp.237-240, 19-23 July 2010.

[4] L. Rui; Z. Hongliang; X. Yang; Y. Yixian; W. Cong; , "Remote NAT Detect Algorithm Based on Support Vector Machine," Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on, vol., no., pp.1-4, 19-20 Dec. 2009.

[5] P. Phaal, Detecting NAT devices using sFlow. http://www.sflow.org/detectNAT/ (last modified: 2009).

[6] T. Miller, Passive OS fingerprinting: Details and techniques. http://www.ouah.org/incosfingerp.htm (last modified: 2005).

[7] R. Beverly. A robust classifier for passive TCP/IP fingerprinting. In Proc. Conference on Passive and Active Measurement (PAM) (2004).

[8] S. M. Bellovin, "A technique for counting natted hosts", In IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, pp. 267–272, New York,NY, USA, 2002, ACM Press.

[9] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers,1993.

[10] V. Krmicek , J. Vykopal , R.Krejci, Netflow based system for NAT detection, Proceedings of the 5th international student workshop on Emerging networking experiments and technologies, December 01-01, 2009, Rome, Italy.

[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reute-mann, and I. H. Witten, "The weka data mining software:An update," SIGKDD Explorations, vol. 11, no. 1, 2009.

[12] R. Alshammari, A. N. Zincir-Heywood, Machine learning based encrypted traffic classification: Identifying ssh and skype, IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–8, 2009.

[13] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers,1993.

[14] G. H. John and P. Langley Estimating Continuous Distributionsin Bayesian Classifiers. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. pp. 338-345, Morgan Kaufmann, San Mateo, 1995.

[15] Netmate.http://www.ipmeasurement.org/tools/netmate/

[16] IETF.http://www3.ietf.org/proceedings/97apr/97aprfinal/xrtftr70.htm.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[18] M. A. Rajab, F.Monrose, A.Terzis, On the impact of dynamic addressing on malware propagation,In Proceedings of the 2006 ACM Workshop on Recurring Malcode, pages 51–56, 2006.