Identification of Hosts behind a NAT Device Utilizing Multiple Fields of IP and TCP

Hanbyeol Park[1], Seung-hun Shin[2], Byeong-hee Roh[3], and Cheolho Lee[4]
[1]Graduate School of Software, [2]University College, [3]Dept. of Computer Engineering, Ajou University, Suwon, Korea
[4]The Attached Institute of ETRI, Daejeon, Korea
Email: {hbyeolp, sihnsh, bhroh}@ajou.ac.kr, cheolholee75@gmail.com

***Abstract*— NAT provides a function to translate private IP addresses into a public IP address. With NAT functions, hosts with private IP addresses can be connected to the Internet, but they are hidden on the Internet. When malicious hosts behind a NAT device attack service providers on the Internet, firewalls attached on the services may detect the attack. After the attacks are detected, they may block all the traffic including the malicious hosts and other normal ones from the NAT, since all the traffic from the NAT have a same source IP address and the hosts behind the NAT cannot be identified individually by them. In this paper, we propose an effective method to identify hosts behind a NAT device by utilizing multiple fields of IP and TCP such as IPID, TTL, SYN flag, and timestamp. The proposed method can identify the number of hosts behind a NAT device, and their OSs with very high accuracy compared to existing works utilizing only one field.**

***Keywords*— NAT, Passive OS fingerprinting.**

I. INTRODUCTION

The NAT (Network Address Translation) is a method to map several private IP addresses into a public IP address associated with port numbers at the transport layer [1]. With NAT devices, hosts with private IP addresses can connect and exchange data with hosts on the Internet.

With the address translation function of NAT, the network topology and hosts' private IP addresses behind a NAT device (NATD) are hidden on the Internet, instead hosts on the Internet acknowledge only one public IP address representing the hosts behind the NATD.

The NAT functions may cause the following security problem: When malicious hosts behind a NATD attack service providers on the Internet, firewalls attached on the services may detect the attacks. After the attacks are detected, they may block all the traffic including the malicious hosts and other normal ones from the NATD, since all the traffic from the NATD have a same source IP address and the hosts behind the NATD cannot be identified individually by them [2]. With the identification of hosts behind NATDs, the above ineffective countermeasure by firewalls can be overcome.

There have been various works to identify hosts behind a NATD [3-8]. Bellovin [3] proposed the method based on IPID (identification field in IPv4 packets) field, which are configured sequential rule. The method can identify hosts behind a NATD simply by observing sets of sequential IPIDs. However, the method can be applied to Windows OSs only, because only the OSs follow the sequential rule.

This work was supported partially by the National Security Research Institute.

In this paper, we propose an effective method to identify hosts behind a NAT device by utilizing multiple fields of IP and TCP such as IPID and TTL, and SYN flag and timestamp, respectively. The proposed method is based on passive fingerprinting approach, and can identify the number of hosts, and their OSs including Windows, Linux and others with very high accuracy compared to existing works using only one field.

II. RELADTED WORKS

For the identification of hosts behind a NATD, various fields of IP and TCP are utilized.

A method based on IPID filed of IPv4 has been proposed [3], and it can distinguish different hosts by observing set of IPID fields according to the sequential rule of the field, even if the observed source port number is same. But it can be applied only to Windows OSs, since Windows OSs implement the sequential rule on IPID only other than random used in Mac OS and OpenBSD, for example, or jumped-sequential ones. The method can't be used in IPv6, because IPID exists only in IPv4.

Methods based on the TCP timestamps and the boot times of machines have been proposed in [4] and [5]. The methods cannot be applied to Windows OSs, because Windows OSs do not use the timestamp option utilized in them usually.

DPI (Deep Packet Inspection) based approaches also have been proposed. Zhao et al. [6] proposed a method to count the number of hosts by inspecting the payload of packets related to sessions of instant messaging applications, since each user runs one instant messaging client on each host usually.

In [7] and [8], various fields of protocol data units (PDUs) at application layer have been utilized. The fields are HTTP user agent strings, HTTP referrers, HTTP session IDs and cookies, user names, chat pseudonyms, and so on. However, if all hosts use the same OS and applications, the methods may count them as one host. In addition, since they investigate application layer information, they depend on the application layer protocols and need larger computational complexity than others utilizing lower layer fields.

III. THE PROPOSED METHOD

In this paper, we use multiple fields of TCP/IP packets, such as IP ID, TTL, TCP source port and timestamp, to improve accuracy of the host identification. Fig. 1 shows the process of the proposed approach. In the first phase, since our approach uses only TCP SYN packet, we check TCP SYN flag of a packet to determine whether it is TCP SYN or not. In the next phase, we estimate OS of hosts through its initial TTL value. And in the last phase, we classify hosts using OS specific method. With this sequence of multiple packet field investigation, we can divide packets into sets of packets that in each set include packets which are supposed to come from a host. It means that we can estimate the number of hosts behind a NAT also.

```
Algorithm 1 Host Classification Algorithm
INPUT: packet
OUTPUT: $L_s$ : A host list of Windows family, $L_t$ : A host list of Linux and Unix
 1:  if protocol is TCP then
 2:      if flags is SYN then
 3:          if ttl is initial TTL value of Windows then
 4:              $L_s \leftarrow$ SourceportIpidMethod(ipid, sourceport, packetarrivetime)
 5:          else if ttl is initial TTL value of Linux and Unix then
 6:              $L_t \leftarrow$ TimestampMethod(tsval, packetarrivetime)
 7:          end if
 8:      end if
 9:  end if
10:  return $L_s$, $L_t$
```

Figure 1. A proposed method

A. OS estimation phase

Table I shows initial TTL values of various well-known OSs. Each OS uses fixed initial TTL value such as 128 for Windows family and 64 for Linux and Mac OS. So we can classify packets according to TTL value.

TABLE I. THE INITIAL TTL VALUES OF VARIOUS WELL-KNOWN OSS[9]

OS	TTL Initial Value
Windows XP	128
Wimdows Vista	128
Windows 7	128
Windwos 8.1	128
Windows 10	128
Linux Kernel	64
Mac OS X	64

B. Host classification phase

We use two methods to classification. One is the method using IP ID and source port number. Other is the method based on timestamp. Table II shows parameters of the proposed method for Windows family. We dynamically generate and manage a host list which has host information such as IP ID, source port, and packet arrive time. When if a packet is supposed to come from new host, then new host item will be generated and attached to the existing host list. After that, the packet will be allocated into the new host list item. Otherwise, if a packet is supposed to come from a host which is already existing in the host list, then the packet will be added to the appropriate existing host list item. The rule to determine whether a packet comes from the same host is as follows. The difference of arrival time between current packet and the last packet of the host list should be within *timelim*. The difference of source port number between current packet and last packet of the list should be within *portgaplim*. Also the difference of IP ID value between current packet and last packet of the list should be within *ipgaplim*. Moreover, the number of the list item should be more than *fsize*. If these four condition is satisfied then we consider both of packets come from the same host.

TABLE II. PARAMETERS OF THE METHOD IN WINDOWS FAMILY[3][10]

Parameter	Value
timelim	6s
portgaplim	3
ipgaplim	600
fsize	50

1) The method based on IP ID and source port number of TCP SYN packet(SourceportIpidMethod)

Source port number of TCP SYN packet and IP ID value have sequential values in Windows. We utilize sets of source port number of TCP SYN packet depending on IP ID value because most router having NAT function do not usually rewrite source port number, if source port numbers of hosts do not overlap each other and source port number of TCP SYN packet increases sequentially in Windows, like IP ID value.

2) The method based on timestamp(TimestampMethod)

TCP timestamp contains a 32-bit subfield *Tsval* which is obtained from timestamp clock[11]. *Tsval* shows a pattern of linear Eq. (1), where *a* is slop depending on OS, *b* is the initial timestamp value and *s* is current time. The value of *Tsval* usually set to zero at boot time. Thus we estimate the number of hosts using these conditions.

$$t(s) = a \bullet s + b \tag{1}$$

IV. EXPERIMENT

A. Experiment environment

Table III shows experiment scenarios. We use two kinds of OSs, Windows 7 and Linux Kernel 3.2, to check the accuracy of our approach when multiple OSs are existing in the same NAT environment.

TABLE III. EXPERIMENTS SCENARIO

Experiment Scenario	OS	Quantity
1-1	Windows 7	14
1-2		35
2-1	Linux Kernel 3.2	8
2-2		25
3	Windows 7	35
	Linux Kernel 3.2	25

B. Experiment result

We evaluate the accuracy, precision and sensitivity to analyze the performance of proposed approach with the following equations[12]. The accuracy Eq. (2) is the proportion of true results (both true positives and true negatives) among the total number of the cases examined. The Precision Eq. (3) is defined as the proportion of the true positives against all the positive results. And the Sensitivity Eq. (4) refers to the ability of correct detection.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \qquad (2)$$

$$\text{Precision} = TP / (TP + FP) \qquad (3)$$

$$\text{Sensitivity} = TP / (TP + FN) \qquad (4)$$

- TP(True Positive): The number of hosts correctly identified
- TN(True Negative): The number of hosts correctly rejected, 0 in this experiment since there is no packet which doesn't allocate into host list item
- FP(False Positive): The number of hosts incorrectly identified.
- FN(False Negative): The number of hosts incorrectly rejected.

TABLE IV. THE ACCURACY, PRECISION AND SENSITIVITY OF EXPERIMENT RESULTS

Approach	Experiment	Accuracy	Precision	Sensitivity
IP ID	1-1	60%	67%	86%
	1-2	48%	54%	82%
	2-1	28%	28%	100%
	2-2	22%	22%	88%
	3	33%	36%	80%
User-agent	1-1	22%	50%	29%
	1-2	10%	40%	12%
	2-1	38%	100%	38%
	2-2	18%	63%	20%
	3	10%	50%	12%
Proposed method	1-1	87%	93%	93%
	1-2	71%	83%	83%
	2-1	100%	100%	100%
	2-2	88%	100%	88%
	3	76%	89%	83%

Table IV shows the accuracy, precision and sensitivity of experiment results. The approach based on IP ID is comparatively inaccurate in Linux because The IP ID implementation of Linux is jump-sequential. For example, a single Linux host has 20 lines of IP ID values if the host has 20 TCP download sessions while single Windows host has only 1 line of IP ID values. Another approach based on user-agent is inaccurate because they may count them as one host where all hosts use the same OS and application. Moreover, this approach cannot be utilized to identify hosts behind a NAT because Microsoft Internet Explorer can change user-agent information. Our approach shows significantly accurate result, and for the Linux case it shows more accuracy. The reason of Windows case shows less accuracy is that we use source port number for the Windows but it can be rewritten by a NAT to distinguish the hosts which use same source port number. However, our method shows relatively high accuracy than the existing approaches based on IP ID or user-agent.

V. CONCLUSION

In this paper, we proposed the approach using multiple fields of TCP/IP packet to identify hosts behind a NAT device. We used not only IP ID but also source port number, TTL(time to live) and timestamp option fields. We evaluated our approach against two different approaches. The results showed that the total average accuracy of our approach was 84.4% and it was at least 46.2% higher than two other approaches. The proposed approach showed higher performance in terms of precision and sensitivity also.

As a future work, we will conduct an experiment to check accuracy of our approach against large scale network environment. And we also plan to escalate the proposed approach to identify active hosts behind a NAT using additional TCP/IP information.

REFERENCES

[1] P. Srisuresh, and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, 2001.

[2] T. M. Gil, and M. Poleto, "MULTOPS: a data-structure for bandwidth attack detection," Proc. 10th USENIX Security Symposium, pp.23-38, 2001.

[3] S. M. Bellovin, "A technique for counting natted hosts," Proc. the 2nd ACM SIGCOMM Workshop on Internet measurment, pp.267-272, 2002.

[4] E. Bursztein, "Time has something to tell us about network address translation," Proc. Nordic Workshop on Secure IT Systems, 2007.

[5] G. Wicherski, F. Weingarten, and U. Meyer, "IP agnostic real-time traffic filtering and host identification using TCP timestamps," Proc. IEEE 38th Conf. in Local Computer Networks(LCN), pp.647-654, 2013.

[6] L. Zhao, M. Zhang, J. Bi, and J. Wu, "Detecting Private Address Space based on Application Layer Information," Proc. 1st IEEE Workshop on Adaptive Policy-based Management in Network Management and Control, 2006.

[7] G. Maier, F. Schneider, and A. Feldmann, "NAT usage in residential broadband networks," Int. Conf. on Passive and Active Network Measurement, 2011.

[8] M. I. Cohen, "Source Attribution for Network Address Translated forensic captures," Digital Investigation, vol.5, no.3-4, pp.138-145, 2009.

[9] R. Tyagi, T. Paul, B. S. Manoj, and B. Thanudas, "Packet Inspection for Unauthorized OS Detection in Enterprises," IEEE Security and Privacy, vol.13, no.4, pp.60-65, 2015.

[10] S. Mongkolluksamee, K. Fukuda, and P. Pongpaibool. "Counting NATted hosts by observing TCP/IP field behaviors," IEEE Int. Conf. Commun. (ICC), 2012.

[11] V. Jacobson, R. Braden, and D. Borman, "TCP extensions for high performance," RFC 1323. 1992.

[12] D. G. Altman, and J. M. Bland, "Statistics Notes: Diagnostic tests 1: sensitivity and specificity," British Medical Journal, vol. 308, pp.1552, 1994.