Security Enhancement by Detecting Network Address Translation based on Instant Messaging

Jun Bi, Miao Zhang, and Lei Zhao

Network Research Center
Tsinghua University
Beijing, P.R.China, 100084 ,
junbi@tsinghua.edu.cn

Abstract. Detecting network address translation is helpful for network administrators to enhance the network security. Current network address translation detection approaches can not work effectively in all scenarios. In this paper, a new detection scheme ImNatDet utilizing instant messaging information is presented, a case study based on characters of MSN Messenger is analyzed, and related security issues are discussed. This paper also indicates that characters of instant messaging applications can be used to detect users' privacy information.

1. Introduction

Mainly due to lack of unallocated IP address in IPv4, network address translation [1] is widely used in today's Internet. The network address translation technologies can be divided into three categories [2]: full cone network address translation, restricted cone network address translation, and symmetric network address translation. Full cone and restricted cone network address translation will map the same external IP address and port from the same internal IP address and port. In symmetric network address translation, the external IP address and port also depend on the destination IP address. According the report in [3], about 17% to 25% of Internet game players are located behind network address translator. We can estimate that the source addresses of a large number of Internet applications are translated by network address translators.

Multiple requirements have been discussed on detection of network address translation [4][5][6]. From the viewpoint of network security, the source address of a packet is replaced by the external address of a network address translator, so it's hard for network administrators to trace back the origin of a packet. For example, if the origin of SPAM or DDoS attacks is located behind a network address translator, then network administrators can only traced the attack packets back to the external boundary of network address translator. Therefore, network address translator can be used by attackers to masquerade their attacking hosts.

Based on the above discussion, monitoring the usage of network address translation will be helpful for service providers to enhance the network security. However, existing network address translation detection approaches can not work effectively in all scenarios.

In this paper, a new detection scheme ImNatDet is proposed, in which the instant messaging information is utilized.

The rest of this paper is organized as follows: in Section 2, related works are introduced; in Section 3, we present the detection algorithm and a case study on MSN

messenger; some related issues are discussed in Section 4; Section 5 summarizes the paper and discusses future work.

2. Related Works

Investigators have proposed several methods to detect network address translation. We summarize related works as four categories according to the information used for detection algorithms: by detecting TTL value in the IP header [7]; by detecting IP ID field in the IP header [8]; by detecting TCP/IP stack implementation information of different operating systems [9][10][11]; by detecting time related information in the TCP header [12].

These methods rely on the information of TCP/IP stack in hosts, so they can be defeated by attackers in modifying TCP/IP stack in their attacking hosts to hide the address translation information.

Another possible way to detect network address translation is checking whether the source port number in TCP/UDP packets coming from a target source address reaches a high number quickly. This is the character that a large scale network is behind a network address translator, or the character of the usual setting on network address translator that using high port number for translated TCP/UDP packets to avoid collision with possible service port numbers on the translator.

This method can be also defeated if the attackers are able to change the settings on the network address translator.

Instant messaging is an emerging and popular communication method for Internet users. Since the first instant messaging tool UNIX “talk” was introduced in 1973, IRC (Internet Relay Chat) was introduced in 1988, and ICQ was introduced in 1996, subsequently there rises many public instant messaging service provides, such as AIM (AOL), MSN Messenger (Microsoft), Google Talk (Google), Yahoo! Messenger (Yahoo), etc. In China, TENCENT is recognized as the most successful instant messaging service provider with its instant messaging tool named QQ. Some enterprises also provide private instant messaging services for business purposes. The characters of real-time communication and presence awareness led to its acceptance as the third popularity Internet application following Email and Web [13]. As an evolving technology, instant messaging is attracting researches in the related research topics.

This paper will study characteristics of instant messaging and utilize these characters for detection on network address translation. The basic idea of instant messaging based network address translation detection is as follows. Instant messaging applications are popular Internet applicant and are user–oriented. Normally user will run only one instance of instant messaging application on a host. If there are more than one instance was found, it means there is more than one host behind the IP address and their original source addresses were changed by the network address translator.

3. Methodology

3.1 Scenario

The most important characteristic of instant messaging is the presence service [14]. Hereafter we use the term ”presence packets” to denote the packets which carry the presence information; use the term “presence channel” to denote the data channel between an instant messaging client and an instant messaging server which transfers presence packets; and use the term “presence channel packets” to denote the packets (including the presence packets) transferred in the presence channel.

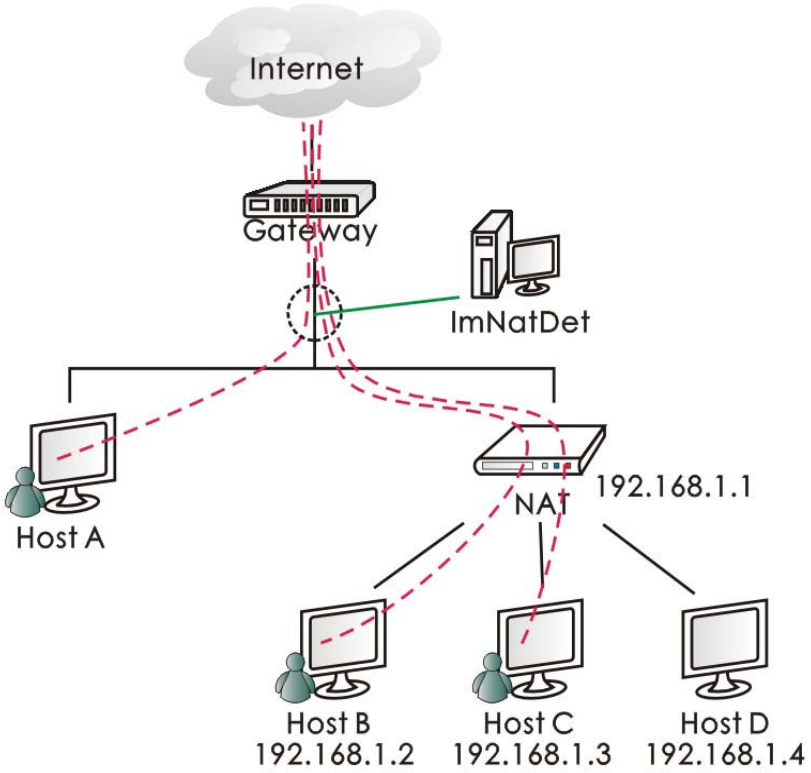

Fig. 1. ImNatDet Detection Scenario

The scenario of instant messaging based detection is shown in figure 1. Host *A*, *B*, *C*, and *D* share the same gateway to access the Internet. Host *B*, *C*, and *D* are in a private address space behind a network address translator. A device running the detection algorithm, which is called ImNatDet, is passively collecting and analyzing IP packets passing through the access gateway.

3.2 Algorithm

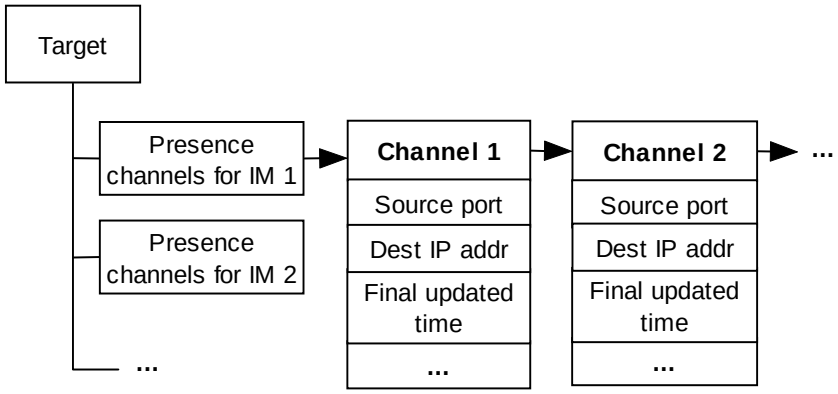

Fig. 2. Presence Channel Table

Figure 2 shows the channel table used for network address translation detection in ImNatDet. For each target IP address in the edge network, a list of presence channels is maintained for each kind of instant messaging application. When a presence channel packet for a specific instant messaging application coming from a given target IP address is captured, ImNatDet updates the presence channel list of that instant messaging application by the information gotten from that packet and counts the number of current presence channels. If it exceeds a threshold N_{th}, ImNatDet would determine that there is a network address translator on this target IP address. Figure 3 shows the processing procedure for captured presence packets

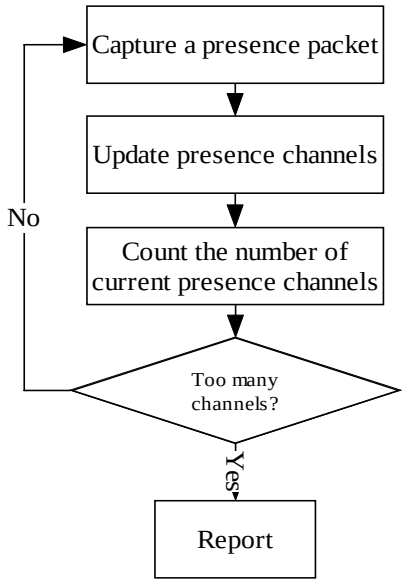

Fig. 3. Presence Packets Processing Procedure

There are two important problems:

(1) How to sort out presence channel packets from all captured packets.

Considering the possible high bandwidth of the access gateway, the filtering method must be simple and efficient.

(2) How to derive the number of presence channels (the number of instances) from the captured packets.

We assumed that there is more than one host running the same instant messaging application behind a network address translator and there is only one instance for each type of instant messaging application allowed on one host.

Different instant messaging applications have different design in their presence channels. We have to implement different filtering mechanisms for different instant messaging applications. In this paper, we choose Microsoft MSN Messenger as the sample instant messaging application to design the algorithm.

In the following part of this section, we will show some observations on MSN Messenger, then introduce the method to sort out presence channel packets. At last, we discuss how to derive the number of presence channels.

3.3 Case Study: MSN Messenger

Currently, there are three major instant messaging protocol suites: IMPP (Instant Messaging and Presence Protocol), XMPP (Extensible Messaging and Presence Protocol), and SIMPLE (SIP for Instant Messaging and Presence Leverage Extension). Some instant messaging applications use private protocols,

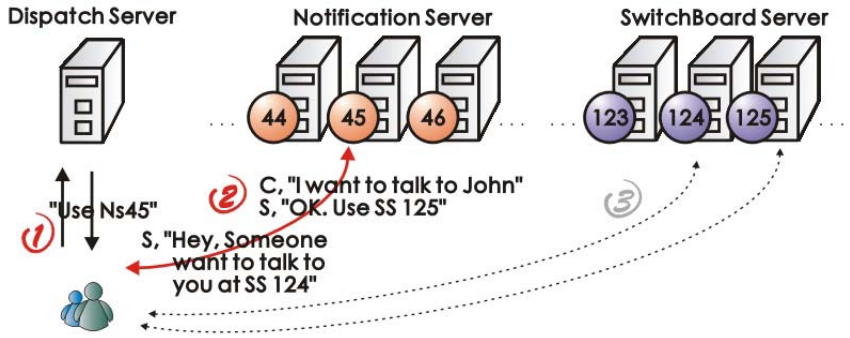

Fig. 4. MSN Messenger

MSN Messenger uses MSN Messenger Service Protocol. Expired IETF draft [15] describes the version 1.0 of the protocol. Some analysis about MSN Messenger can be found it [16] and [17].

The mechanism of MSN Messenger is shown in figure 4:

(1) The MSN Messenger client sets up a TCP connection to the dispatch server at destination port 1863.

(2) The dispatch server dispatches a notification server for this MSN Messenger client.

(3) The MSN Messenger client sets up a TCP connection to the assigned notification server at destination port 1863. Usually, this connection will last for the whole log-on session, except the case that the notification server tells the MSN

Messenger client to connect to another notification server when it is overloaded or is about to be shutdown for maintenance .

(4) When the MSN Messenger client needs to send instant messages or transfer files to other MSN Messenger clients, it asks notification server to dispatch a switchboard server to the MSN Messenger client.

(5) The MSN Messenger client sets up a TCP connection to the assigned switchboard server at destination port 1863. The connection between MSN Messenger client and switchboard server will be closed when the chatting is over.

The TCP connection between the MSN Messenger client and the notification server can be considered as the presence channel. The MSN Messenger client periodically sends a ``PNG'' command to the notification server.

3.4 Capturing Presence Channel Packets

Methods that capture presence channel packets by filtering IP addresses of instant messaging servers are efficient, since ImNatDet only captures and examines the first few bytes of an IP packet. But this method requires a full list of IP addresses of instant messaging servers, which is difficult to get for some instant messaging applications.

Methods that capture presence channel packets by filtering TCP/UDP service port numbers of instant messaging servers is also efficient. But usually it can not be used alone because the same port number can be used or both presence channels and other TCP/UDP channels.

Methods that capture presence channel packets by filtering the packet payload for special control commands are less efficient, because ImNatDet has to exams the packet content. But it is a feasible way when the server address based and service port number based methods can't work for some instant messaging applications.

Different techniques are implemented in ImNatDet to capture presence channel packets. To get the presence channel packet of MSN Messenger, we apply service port number and payload characteristic as the packet filtering criteria.

From the observation on MSN Messenger mechanism discussed in last section, we found that TCP packets between the target being detected and one of the notification servers are what we want to sort out. Since there are fairly a large number of notification servers and it is difficult to collect all addresses of notification servers, we did not use server address as the filtering condition. We filter presence channel data of MSN Messenger by checking whether the destination port number is 1863 and whether there is a string "PNG" in the payload.

3.5 Counting the Number of Instants

For each kind of instant messaging application, a list of presence channel (represented by destination IP and source port) is maintained for each target source IP address of local network; and a timestamp T_f is set for each channel to denote the final updated time of that channel. When a presence channel packet is captured, we update the flow list and check the number of the list as below:

(1) Get the destination IP address and source port number and check whether it belongs to an existing flow. If it does, update T_f of the flow; otherwise, append a new record to the list.

(2) Remove the flow records that haven't been updated for T_{max} time. T_{max} is the maximum value of the time interval between two captured packets for one presence channel.

(3) Count the number of concurrent flows and check whether it is larger than a threshold N_{th} to determine whether the target is a network address translator.

There are also some instant messaging applications that use UDP in transmitting presence information. For these instant messaging applications, as long as the client port number is relative stable and network address translator doesn't use different port number to transfer UDP packets for the same presence channel, these UDP based presence packets can be processed in the same algorithm as TCP packets.

4. Related Issues

4.1 Design Considerations

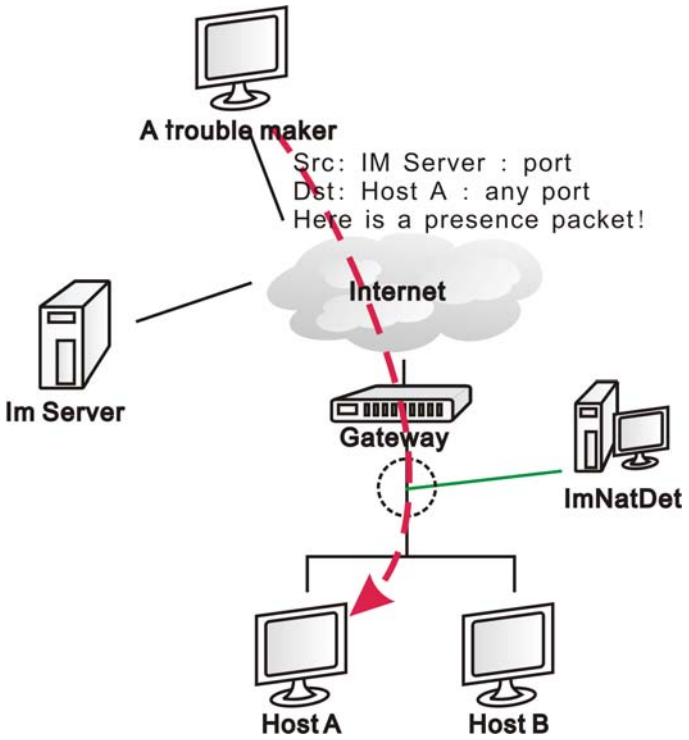

Fig. 5. An Attack Scenario

We can passively capture both incoming and outgoing packets passing through the detection point. In the design of ImNatDet, we choose to only capture the outgoing traffics for detection, so that we can prevent possible attack on our detection method from the outside.

As shown in figure 5, a trouble maker outside can spoof the IP address of an instant messaging server and periodically send forged instant messaging presence packets to two ports of host *A*, which a normal host in the local network. If ImNatDet

uses these incoming packets to update the presence channels table, it would consider host *A* as a network address translator.

Though it is possible for hosts inside the local network to spoof IP addresses of other local hosts and send forged presence packet to instant messaging servers, the source of these packets are easy to be tracked back.

4.2 Correctness

ImNatDet is based on the assumption that instant messaging client always keeps a stable data channel to its server. It is also possible that instant messaging does not behave in this way. For example,

(1) An UDP based instant messaging application may change source port number of a presence channel;

(2) A TCP based instant messaging applications may keep the presence channel for a short time: it opens a TCP connection when it needs to send a presence packet, then close the TCP connection after the presence packet is sent out.

The ImNatDet detection will fail on instant messaging applications using the above mechanism. We did investigations on current popular instant messaging applications and we haven't found such a case yet.

The proxy is also a popular way to provide network address translation. Most instant messaging applications support login through a socks proxy or a HTTP proxy. ImNatDet can also detect socks proxy [18] in a local network. When clients use instant messaging via a socks proxy inside the monitored local network, presence packets are transferred via this proxy. Since the socks proxy simply relays the data of the transport layer, the characteristics of presence packets will be kept, so ImNatDet would detect these presence channels coming from the sock proxy.

Because packets are encapsulated by HTTP proxy, ImNatDet can not capture presence packets without original TCP information. Therefore, ImNatDet can not be used to detect local HTTP proxy.

4.3 Privacy Issues of Instant Messaging

The detection results of ImNatDet also expose some users' privacy information on how to use instant messaging in the local network:

(1) The type of instant messaging applications being used.

(2) The frequency of instant messaging applications being used.

(3) The behaviors such as the start and end time of an instant messaging application being used.

It means that the characters used by network address translation detection can be also utilized by hackers to detect user's privacy information. A hacker can even use the characters to sort out instant messaging packets and capture the communication contents. Considering instant messaging is widely used in today's business communications, this problem should be seriously investigated.

5. Conclusion

This paper proposed a new network address translation detecting scheme ImNetDet by utilizing instant messaging information. The proposed method can help network administrators to enhance the network security.

Currently there is no detection method that can work well in every scenario. When a detecting method is put forward, a corresponding anti-detecting method is proposed. Thus it is required to combine multiple detection methods to get the better results.

This paper also reveals that some characters of instant messaging applications might be used to detect users' privacy information. Instant messaging application designers should consider how to design trustworthy instant messaging mechanisms. One possible direction is to avoid using fixed port numbers to transmit presence information.

Proxy is another popular mechanism to provide network address transition and sometimes brings network management and security problems. ImNatDet can be directly used to detect socks proxy in the local network, but can not be used to detect a local HTTP proxy. For the future work, the passive detection on local HTTP proxy is necessary to be investigated.

References

1. Srisuresh P. and Egevang K.: Traditional IP Network Address Translator (Traditional NAT), RFC3022, Jan 2001.
2. Rosenberg, J., Weinberger, J., and Huitema, C.: STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), March 2003
3. Armitage, G.: Inferring the Extent of Network Address Port Translation at Public/Private Internet Boundaries, Technical Report 020712A, CAIA, 2002.
4. Hain, T.: Architectural Implications of NAT, RFC2993, Nov 2000.
5. Holdrege, M. and Srisuresh, P.: Protocol Complications with the IP Network Address Translator, RFC3027, Jan 2001.
6. Senie, D.: Network Address Translator (NAT)-friendly Application Design Guidelines, RFC3235, Jan 2002.
7. Phaal, P.: Detecting NAT devices Using sFlow, http://www.sflow.org/detectNAT, 2003.
8. Bellovin, S.: A Technique for Counting NATted Hosts, the 2nd Internet Measurement Workshop, Nov 2002.
9. Zalewski, M.: Passive OS Fingerprinting Tool, http://lcamtuf.coredump.cx/ p0f.shtml, 2003.
10. Kaniewski, W.: Detect NAT Users in Your LAN, http://toxygen.net/misc/, 2000.
11. Ulikowski, M.: NAT Detection Tool, http://elceef.itsec.pl/natdet/, 2003.
12. Kohno, T., Broido, A., Claffy, K.: Remote Physical Device Fingerprinting, IEEE Symposium on Security and Privacy, 2005.
13. Isaacs, E., Walendowski, A., Whittaker, S., Schiano, D., Kamm, C.: The Character, Functions, and Styles of Instant Messaging in the Workplace, CSCW '02, New Orleans, Louisiana, USA, Nov 2002,.
14. Day, M., Rosenberg, J., and Sugano, H.: A Model for Presence and Instant Messaging, RFC2778, Feb 2000.
15. Movva, R.: MSN Messenger Service 1.0 Protocol, draft-movva-msn-messenger-protocol-00.txt, Aug 1999.
16. MSN Messenger Protocol, http://www.hypothetic.org/docs/msn/.
17. MSNPiki, Unofficial MSN Protocol Documentation, http://msnpiki.msnfanatic. com/
18. Leech, M.: SOCKS Protocol V5, RFC 1928, Mar 1996.