Time has something to tell us about Network Address Translation

Elie Bursztein*

Abstract

In this paper we introduce a new technique to count the number of hosts behind a NAT. This technique based on TCP timestamp option, works with Linux and BSD system and therefore is complementary to the previous one base on IPID than does not work for those systems. Our implementation demonstrates the practicability of this method.

1 Introduction

Network Address Translation (NAT) also known as IP Masquerading involves re-writing the source and/or destination addresses of IP packets as they pass through a router or firewall. Most systems using NAT in order to enable multiple hosts on a private network to access the Internet using a single public IP address. This technique has became popular for number of reasons, including the shortage of IPv4 addresses. It has become a standard feature in routers for home and small-office Internet connections. Until now, the only method known to count how many host are hidden behind a NAT rely on the IPID field. However, this method has main limitation: it does not work with systems that does not implement the IPID field as a simple counter. For instance Linux uses the constant 0 for the IPID field FreeBSD and OpenBSD use a pseudo random generator for the IPID field. This is why a complementary approach is necessary to deal with these implementations.

*LSV, ENS Cachan, CNRS, INRIA, PhD student, supported by a DGA grant. eb@lsv.ens-cachan.fr

The main contribution of this paper is a technique based on the analysis of the TCP timestamp option to count the hosts behind a NAT. Experimentation shows that this technique works against Linux and OpenBSD NAT mechanism. Additionally, this technique can be use to performs active fingerprint consistency.

The remainder of this paper is structured as follows. Section 2 provides an overview of related work concerning NAT detection. Section 3 presents the TCP timestamp option and its current use in security. Section 4 details the algorithm that use the timestamp for NAT counting. Section 5 introduces several additionally use of the techniques. Section 6 covers the limitation of the technique and how it can be evaded. Section 7 concludes the paper discussing directions for future work.

2 Related Work

The NAt mechanism was introduced in [4], and the TCP timestamp option in [6]. The technique that use IPID field to count NATed hosts was presented in [1]. The IPID field was designed to use in fragment reassembly. RFC 791 [?] describes its role as follows:

The identification field is used to distinguish the fragments of one datagram from those of an- other. The originating protocol module of an internet datagram sets the identification field to a value that must be unique for that source-destination pair and protocol for the time the datagram will be active in the internet system. The originating protocol module of a complete datagram sets the more-fragments flag to zero and the fragment offset to zero.

Several improvement were made to this technique including the use of OS fingerprinting as in [3]. NAT can also been detected because it may alters some packet headers as suggested in [9]. The timestamp option is already used in security for uptime analysis [5]. The first use of IPID simple counter behavior, for port scanning was presented in [2].

3 The TCP Timestamp Option

The Timestamps option carries two four-byte timestamp fields. The timestamp value field (TSval) contains the current value of the timestamp clock of the TCP sending the option. According to RFC1323: *"The timestamps are used for two distinct mechanisms: RTTM (Round Trip Time Measurement) and PAWS (Protect Against Wrapped Sequences).* Each OS implementation increments this value at a specific rate. So it is possible to infers the system uptime from the timestamp option if the targeted OS is known [7]. For example Linux increments it every 1ms and and FreeBSD every 500 ms. This technique is used in the popular scanner Nmap used to infer remotely the computer uptime :

```
Uptime 26.271 days (since Wed Jun 27 09:31:13 2007)
```

Once Nmap has determined the OS type by the mean of TCP/IP fingerprinting, it computes the uptime using the following formula:

$$\frac{timestamp}{numincbysec} = uptimeinsec$$

This is the simplified version that does not take into account the TSval wrap around. The OS specific increment used in our algorithm are those found in Nmap source code.

4 Counting Hosts using Timestamp

The timestamp value can be represented as a linear function:

```
timestamp = numincbysec * sec + in
```

where *numincbysec* is the slope and is dependent of the OS and *in* is the initial timestamp value. The initial number is zero for every OS type except windows. *in* is random for Windows computer, thus the uptime cannot be computed remotely.

To overcome this limitation and have a technique that can be used passively we do not rely on the OS fingerprinting to infer the increments or the initial timestamp value. Instead we use two values of the timestamp, to solve the following equation:

$$ts1 - ts2 = A * (s1 - s2) \Leftrightarrow A = \frac{ts1 - ts2}{s1 - s2}$$

where $ts1$ and $ts2$ are the two timestamp gathered at time $t1$ and $t2$. This allows us to determine passively the increment.

The NAT counting algorithm is based on the idea that two hosts behind a NAT have a distinctive timestamp because they can't have the same uptime unless they have been booted within the same millisecond which is more than unlikely. Intuitively the counting algorithm works as follows: For each host a set of linear functions is stored. Every packet timestamp is verified against this set of linear functions. If it belongs to one of the already known functions then the host is already counted. If not then a new host is detected.
This algorithm is efficient in term of space complexity because it requires only to keep two points for each real host. It is also efficient in time complexity because it only works with simple linear function. As additional benefit, knowing the increment for a given host allows to infer it type of OS. This allows to fingerprint computer that are hidden behind a NAT.
Experiment has shown that it works against standard router such as Netgear one but also against OpenBSD scrub NAT function because this function does not modify the timestamp option.

5 Additional Uses

Another application of the method is to combine it with other active fingerprinting methods. This is useful to defeat anti-fingerprint, by comparing the computed increment with the OS type detected by the fingerprint active method. Of course the method can be used by it self as a fingerprint method but it is less accurate since it can only infer the type of OS.

PAT (Port Address Translation) is used to access a service which is located on computer behind a NAT. Once again our method can help to determine if a PAT mechanism is used because the linear function associated to the PAT will not be consistent with the router one. Moreover if the PAT is used for round robin purpose then it became possible to detect it and infer the number of hosts used in the round robin. There will be one linear function for each host that participate to the round robin. To the best of our knowledge this is the only technique usable to determine how many hosts are involved in a PAT round robin.

6 Limitation and Evasion

The main limitation of the technique is the fact that by default Windows system does not use the timestamp option when they establish a connection [8]. This limitation only affects the passive analysis because if the active probe uses the timestamp option, then the Windows system response will also contains the timestamp option. As explained in the introduction, the host counting technique based on the IPID works with Windows system, hence the combination of the two techniques allows to overcome this limitation. To be complete on the subject, it seems that Windows 95, 98, Me TCP/IP stack does not implement the timestamp option. The other limitation is due to the network latency: it may introduce a bias in the measurement of the time interval between two packets. Fortunately the increment value used by the different class of OS are sufficiently different to allow the implementation to have a window of acceptance large enough to overcome this limitation. Experiment confirms that on a production network, this measure is sufficient. There is mainly three methods to evade this technique:

1. Disabling the TCP timestamp option. This is not a viable option since it is needed on high speed network in particular to prevent sequence wrap around problem.
2. Add a rewriting mechanism to the NAT computer in order to normalize packets timestamp.
3. Change the OS implementation so it use a random increment value for its timestamp.

7 Conclusion

In this paper we have introduced a new method to count hosts hidden behind NAT. This method uses the TCP timestamp option. We also have presented how this method can be used to infers the OS type of each computer hidden behind a given NAT and how to use it to improve active fingerprinting. A future work is to see how this method can be combined to other passive fingerprint methods to improve their accuracy.

References

[1] *A Technique for Counting NATted Hosts*, 2002.

[2] antirez. dumbscan. Technical report, Bugtrack, Dec 1998.

[3] R. Beverly. A robust classifier for passive tcp/ip fingerprinting. In *Passive and Active Network Measurement*, volume 3015 of *LNCS*, pages 158–167. Springer-Verlag, 2004.

[4] K. Egevang, Cray Communications, and P. Francis. Rfc 1631 : The ip network address translator (nat). Technical report, IETF, 1994.

[5] Fyodor. Nmap : free open source utility for network exploration or security auditing.

[6] Jacobson, Braden, and Borman. rfc1323: Tcp extensions for high performance. Technical report, IETF, 1992.

[7] B. McDanel. Tcp timestamping - obtaining system uptime remotely. Technical report, Bugtrack, Mar 2001.

[8] microsoft. Tcp extensions for high performance in windows. knowledge base.

[9] Michal Zalewski. P0f2: "dr. jekyll had something to hyde" passive os fingerprinting tool. Web, 2006.