Effects of port randomization with TCP TIME-WAIT state.
draft-ananth-tsvwg-timewait-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 7, 2009.
Abstract

Source port randomization has been suggested to provide improved security and obfuscation which helps in adding robustness towards blind attacks. With TCP in practice, simply producing a random port as the source port for a new connection can lead to problems when a TCP client establishes connections to a TCP server at a high rate. If the same source port value is chosen twice, the client TCP connection can fail due to the server having the Transmission Control Block (TCB) for this tuple lingering in TIME-WAIT state.

This memo discusses the ramifications of such source port reuse scenarios and suggests some mitigations to avoid the same.

Table of Contents

1. Introduction .................................................. 3
2. The TIME-WAIT problem ...................................... 4
3. Approaches to avoid issues due to port randomization ...... 6
   3.1. Recommended solution ................................. 6
   3.2. Implementation methods ............................. 6
       3.2.1. Method 1: Bit Array and Timers .............. 6
       3.2.2. Method 2: Bit Array and Passive Timers .... 7
       3.2.3. Method 3: 2-Bit Array and Timers ........... 7
4. Acknowledgments ............................................. 9
5. Informative References .................................... 10
Authors’ Addresses ............................................. 11
Intellectual Property and Copyright Statements ................ 12
1. Introduction

The draft [I-D.ietf-tsvwg-port-randomization] recommends that transport protocols SHOULD allocate ephemeral port numbers randomly, since this improves obfuscation of ephemeral port numbers and thus, for relatively little cost, improves susceptibility to blind attacks.

Choosing random ephemeral ports can lead to problems when TCP clients establish connections to TCP servers at a high rate or with a small ephemeral port range. If an ephemeral port is randomly chosen twice before the TCP server has closed a TCB in TIME-WAIT for the port, this may result in connection failure or other side effects.

The subsequent sections of this document cover the issue in detail and discuss solutions to the issue. The methods described in this document are intended to complement and not replace those described in [I-D.ietf-tsvwg-port-randomization].
2. The TIME-WAIT problem

When a TCP connection is actively closed by the server, it will keep the TCB of the connection around in TIME-WAIT state for 2*MSL. The passive closer (client) removes its TCB altogether (once it gets the ACK for the FIN) for the connection and keeps no state. When the client attempts a new connection to the server, if the random source port selection algorithm returns the same source port as the previous incarnation of this connection within the 2*MSL time period, one of the following things can happen:

a) TIME-WAIT assassination of the server TCB. As per RFC 793, if the SYN is in window, the server would respond with RST. If the SYN is outside the window, an ACK would be sent back which would solicit an RST back.

b) As per RFC 1122, some implementations would allow the SYN to be accepted to reopen the connection directly from TIME-WAIT state, if the sequence number of the SYN is larger than the largest sequence number it used on the previous connection incarnation.

c) The SYN would be dropped by the server to prevent TIME-WAIT assassination. This would cause the connection attempts from the client to stall until the server TCB is destroyed.

It needs to be noted that TCP permits a condition known as simultaneous close where both ends send FIN and therefore end up in TIME-WAIT state. In such cases, since the source port selection is a local decision, the new connection attempt can be made by choosing a different source port than the previous incarnation. Of further note, if the client initiates the FIN and thus is the active closer of the connection, the client will retain a TCB in TIME-WAIT state, such that the random port algorithm should not be able to return this port as available.

a) and c) can be attributed to the side effects of port randomization. The behavior b) cannot be guaranteed to be implemented everywhere and also it depends on the sequence number which may be chosen randomly as well. There are a few factors that can lead to scenarios described above occurring more readily:

- The Random Number Generator(RNG) is poor leading to repeated values often. It needs to be noted that with a stateless RNG, frequent repetitions will occur (the birthday paradox).
- The ephemeral port range that the client is utilizing is small and not taking advantage of the large recommended port space which is 1024-65535. Moreover a host or a router can have many TCP applications running, which can result in active as well as passive connection opens which in turn puts a limit on the port availability.

- The client is connecting to the server at a high rate, bringing up and tearing down connections quickly with respect to 2*MSL time.

It is important to note that many applications (e.g., voice applications, RTP, etc.) are not designed to be able to take advantage of the full ephemeral port range and are required by the design characteristics of the application to only use a subset of these ports. Even if all applications are able to take advantage of the full recommended ephemeral port range (1024 to 65535) and are able to utilize a very good random number generator, the connection rate can be high enough to lead to this TIME-WAIT problem.

One of the main motivations to write this document stems from the fact that this issue was observed in the field with one of our voice gateway products. The running code had the TCP/IP stack where the source ports were randomized. The client application was the Interactive Voice Response (IVR) HTTP client which fetches VXML documents from a VXML server (Windows 2003) and does audio playback. When the connection load increased, many server connections were left hanging around in TIME-WAIT state. The port randomization algorithm on the client sometimes returned repeated port values, which resulted in the server dropped the SYN packets. This indirectly affected the connection rate and caused users to view this as a failure of the IVR application. This problem did not occur in earlier versions of the TCP/IP stack which did not include port randomization but instead returned sequential ports.
3. Approaches to avoid issues due to port randomization

3.1. Recommended solution

The solution lies in the client to avoid reusing the same source port for a duration of the server’s TIME-WAIT state after it has passively closed the connection. Since many servers may have varying TIME-WAIT timeouts, it is recommended that length of the timer SHOULD be 2*MSL or 4 minutes. It needs to be noted that the TCB and connection resources can be released and only the source port would be marked unusable for this duration.

3.2. Implementation methods

The following subsections list some possible implementation methods for this issue.

3.2.1. Method 1: Bit Array and Timers

One approach to solving this problem is to utilize a bit array representing each possible port. Each bit represents one of the ephemeral ports 1024 to 65535. If the bit is 0, the port is available for use and if the bit is 1, the port is unavailable for use.

When a connection is opened using an ephemeral port, the bit is set to 1 in the bit array. When a connection is actively closed and transitions from TIME-WAIT to CLOSED and the TCB is removed, the bit representing the port is returned to 0. When a connection is passively closed, no change is made to the bit array. Instead, a timer is started for 2*MSL, and when it expires, the bit representing this port number in the bit array is returned to 0.

The algorithm for generating random port numbers can then consult the bit array before returning a port as available. If it is unavailable, find another port number instead to try. Any of the 4 algorithms for finding ports described in [I-D.ietf-tsvwg-port-randomization] can still be used with this approach. This approach simply means the check:

if (five-tuple is unique)

includes consulting the bit array of ports to determine if a port is available.

It is also important to note that this easily can be the same bit array referred to in [I-D.ietf-tsvwg-port-randomization] to keep the user-specific server ports from being returned by the random port
number algorithm. The user-specific server ports can be reserved using the bit array by setting the bits corresponding to the user-specific server ports to 1. E.g., the setting of these bits could be performed when the listening port is opened.

3.2.2. Method 2: Bit Array and Passive Timers

This is same as the above except with the following variation. The timer mentioned could be a passive timer, in the sense that instead of having an active timer running for each port, we could simply record the timestamp during connection closure. A recurring timer (background process) which awakens every 2*MSL would scan the array and release all the ports whose timestamps have elapsed are >= 2*MSL from the recorded timestamp. This has the drawback that the port reuse duration is indeterministic (i.e. somewhere between 2*MSL and 4*MSL)

Another slight variation of the above scheme would be to not have timer (background process) at all. This is an on-demand approach, which relies on checking whether the source port satisfies the condition or not (i.e. 2*MSL has elapsed or not) at the time the port request happens (during bind() or implicit bind()). The issue with this scheme is that you will have the array maintained for a longer period of time and also extra latency incurred to make sure whether the port is safe to be re-used.

3.2.3. Method 3: 2-Bit Array and Timers

The cost of implementing the approach described in 3.1 is relatively low in terms of memory space for the bit array (~8KBytes), but may possibly require a timer running for all 64535 ephemeral ports. An alternative approach is to utilize a 2-bit array and a single timer or sleeping (background) process.

Two bits in the array are used to represent state for each port. A bit setting of 00 indicates the port is available for use. A bit setting of 01 indicates a port is in use. When a connection is actively closed and transitions from TIME-WAIT to CLOSED and the TCB is removed, the bits representing the port are restored to 00. When a connection is passively closed, the bits representing the port are set to 10.

A single timer or sleeping process can be used to expire or awaken at an interval of 2*MSL. When this timer event occurs or the process awakens, all ports in the table with bits set to 10 are set to 11, indicating the ports are ready to be released. All ports in the table with bits set to 11 are then set to 00, and these ports are now available for use.
This approach is similar to the approach mentioned in 3.2 except that it uses bit array to store the port state instead of having to store a timestamp which denotes the port release time.
4. Acknowledgments

Thanks to James Polk for his useful suggestions. Thanks to Jason Kuo for his insight into the workings of the IVR client application. Special thanks to Siva Yaragalla for his comments and suggestions on various aspects of this draft.
5. Informative References

[I-D.ietf-tsvwg-port-randomization]
Larsen, M. and F. Gont, "Port Randomization",
draft-ietf-tsvwg-port-randomization-01 (work in progress),
February 2008.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7,
RFC 793, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Authors’ Addresses

Anantha Ramaiah
Cisco Systems
170 Tasman Drive
San Jose, CA  95134
USA
Phone: +1 (408) 525-6486
Email: ananth@cisco.com

Patrick Tate
Cisco Systems
4055 Faber Place Drive
Suit 100
North Charleston, South Carolina  29405
USA
Phone: +1 (678) 352-2730
Email: ptate@cisco.com
Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.