On the implementation of the TCP urgent mechanism
draft-ietf-tcpm-urgent-data-03.txt

Abstract

   This document analyzes how current TCP implementations process TCP
   urgent indications, and how the behavior of some widely-deployed
   middle-boxes affect how urgent indications are processed by end
   systems. This document updates the relevant specifications such that
   they accommodate current practice in processing TCP urgent
   indications, provides advice to applications that make use of the
   urgent mechanism, and raises awareness about the reliability of TCP
   urgent indications in the current Internet.

Status of this Memo

   This Internet-Draft is submitted to IETF in full conformance with the
   provisions of BCP 78 and 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 August 23, 2010.

Copyright Notice

Table of Contents

1. Introduction

   This document analyzes how some current TCP implementations process
   TCP urgent indications, and how the behavior of some widely-deployed
   middle-boxes affect the processing of urgent indications by hosts.
   This document updates RFC 1122 [RFC1122] such that it accommodates
   current practice in processing TCP urgent indications, provides
   advice to applications using the urgent mechanism, and raises
   awareness about the reliability of TCP urgent indications in the
   current Internet.

   Section 2 describes what the current IETF secifications state with
   respect to TCP urgent indications. Section 3 describes how some
   current TCP implementations actually process TCP urgent indications.
   Section 4 updates RFC 1122 [RFC1122] such that it accommodates
   current practice in processing TCP urgent indications. Section 5
   provides advice to to new applications employing TCP, with respect to
   the TCP urgent mechanism. Section 6 provides advice to existing
   applications that use or rely on the the TCP urgent mechanism.

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119 [RFC2119].

2. Specification of the TCP urgent mechanism

2.1. Semantics of urgent inications

   TCP incorporates an "urgent mechanism" that allows the sending user
   to stimulate the receiving user to accept some "urgent data" and to
   permit the receiving TCP to indicate to the receiving user when all
   the currently known urgent data have been received by the user.

   The TCP urgent mechanism permits a point in the data stream to be
   designated as the end of urgent information. Whenever this point is
   in advance of the receive sequence number (RCV.NXT) at the receiving
   TCP, that TCP must tell the user to go into "urgent mode"; when the
   receive sequence number catches up to the urgent pointer, the TCP
   must tell user to go into "normal mode" [RFC0793]. This means, for
   example, that data that were received as "normal data" might become
   "urgent data" if an urgent indication is received in some successive
   TCP segment before those data are consumed by the TCP user.

   The URG control flag indicates that the "Urgent Pointer" field is
   meaningful and must be added to the segment sequence number to yield
   the urgent pointer. The absence of this flag indicates that there is
   no urgent data outstanding [RFC0793].

The TCP urgent mechanism is NOT a mechanism for sending "out-of-band" data: the so-called "urgent data" should be delivered "in-line" to the TCP user.

2.2. Semantics of the Urgent Pointer

There is some ambiguity in RFC 793 [RFC0793] with respect to the semantics of the Urgent Pointer. Section 3.1 (page 17) of RFC 793 [RFC0793] states that the Urgent Pointer "communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field is only be interpreted in segments with the URG control bit set". However, Section 3.9 (page 56) of RFC 793 [RFC0793] states, when describing the processing of the SEND call in the ESTABLISHED and CLOSE-WAIT states, that "If the urgent flag is set, then SND.UP <- SND.NXT-1 and set the urgent pointer in the outgoing segments".

RFC 961 [RFC0961] clarified this ambiguity in RFC 793 stating that "Page 17 is wrong. The urgent pointer points to the last octet of urgent data (not to the first octet of non-urgent data)". RFC 1122 [RFC1122] formally updated RFC 793 by stating, in Section 4.2.2.4 (page 84), that "the urgent pointer points to the sequence number of the LAST octet (not LAST+1) in a sequence of urgent data."

2.3. Allowed length of urgent data

RFC 793 [RFC0793] allows TCP peers to send urgent data of any length, as the TCP urgent mechanism simply provides a pointer to an interesting point in the data stream. In this respect, Section 4.2.2.4 (page 84) of RFC 1122 explicitly states that "A TCP MUST support a sequence of urgent data of any length".

3. Current implementation practice of TCP urgent data

3.1. Semantics of urgent indications

As discussed in Section 1, the TCP urgent mechanism simply permits a point in the data stream to be designated as the end of urgent information, but does NOT provide a mechanism for sending out of band data.

Unfortunately, virtually all TCP implementations process TCP urgent data differently. By default, the last byte of "urgent data" is delivered "out of band" to the application. That is, it is not delivered as part of the normal data stream. For example, the "out of band" byte is read by an application when a recv(2) system call

with the MSG_OOB flag set is issued.

Most implementations provide a socket option (SO_OOBINLINE) that
allows an application to override the (broken) default processing of
urgent data, so that they are delivered "in band" to the application,
thus providing the semantics intended by the IETF specifications.

3.2. Semantics of the Urgent Pointer

All the popular implementations that the authors of this document
have been able to test interpret the semantics of the TCP Urgent
Pointer as specified in Section 3.1 of RFC 793. This means that even
when RFC 1122 officially updated RFC 793 to clarify the ambiguity in
the semantics of the Urgent Pointer, this clarification never was
reflected into actual implementations (i.e., virtually all
implementations default to the semantics of the urgent pointer
specified in Section 3.1 of RFC 793).

Some operating systems provide a system-wide toggle to override this
behavior, and interpret the semantics of the Urgent Pointer as
clarified in RFC 1122. However, this system-wide toggle has been
found to be inconsistent. For example, Linux provides the sysctl
"tcp_stdurg" (i.e., net.ivp4.tcp_stdurg) that, when set, supposedly
changes the system behavior to interpret the semantics of the TCP
Urgent Pointer as specified in RFC 1122. However, this sysctl
changes the semantics of the Urgent Pointer only for incoming
segments, but not for outgoing segments. This means that if this
sysctl is set, an application might be unable to interoperate with
itself if both the TCP sender and the TCP receiver are running on the
same host.

3.3. Allowed length of urgent data

While Section 4.2.2.4 (page 84) of RFC 1122 explicitly states that "A
TCP MUST support a sequence of urgent data of any length", in
practice all those implementations that interpret TCP urgent
indications as a mechanism for sending out-of-band data keep a buffer
of a single byte for storing the "last byte of urgent data". Thus,
if successive indications of urgent data are received before the
application reads the pending "out of band" byte, that pending byte
will be discarded (i.e., overwritten by the new byte of urgent data).

In order to avoid urgent data from being discarded, some
implementations queue each of the received "urgent bytes", so that
even if another urgent indication is received before the pending
urgent data are consumed by the application, those bytes do not need
to be discarded. Some of these implementations have been known to
fail to enforce any limits on the amount of urgent data that they

queue, thus resulting vulnerable to trivial resource exhaustion
attacks [CPNI-TCP].

It should be reinforced that the aforementioned implementations are
broken. The TCP urgent mechanism is not a mechanism for delivering
out-of-band data.

3.4. Interaction of middle-boxes with TCP urgent indications

As a result of the publication of Network Intrusion Detection (NIDs)
evasion techniques based on TCP urgent indications [phrack], some
middle-boxes clear the urgent indications by clearing the URG flag
and setting the Urgent Pointer to zero. This causes the "urgent
data" to become "in line" (that is, accessible by the read(2) call or
the recv(2) call without the MSG_OOB flag) in the case of those TCP
implementations that implement the urgent mechanism as out-of-band
data (as described in Section 3.1). An example of such a middle-box
is the Cisco PIX firewall [Cisco-PIX]. This should discourage
applications from depending on urgent indications for their correct
operation, as urgent indications may not be not reliable in the
current Internet.

4. Updating RFC 1122

Considering that as long as both the TCP sender and the TCP receiver
implement the same semantics for the Urgent Pointer there is no
functional difference in having the Urgent Pointer point to "the
sequence number of the octet following the urgent data" vs. "the last
octet of urgent data", and since all known implementations interpret
the semantics of the Urgent Pointer as pointing to "the sequence
number of the octet following the urgent data", hereby we update RFC
1122 [RFC1122] such that "the urgent pointer points to the sequence
number of the octet following the urgent data" (in segments with the
URG control bit set), thus accommodating virtually all existing TCP
implementations.

5. Advice to new applications employing TCP

As a result of the issues discussed in Section 3.2 and Section 3.4,
new applications SHOULD NOT employ the TCP urgent mechanism.
However, TCP implementations MUST still include support for the
urgent mechanism such that existing applications can still use it.

[6](#). Advice to applications that make use of the urgent mechanism

   Even though applications SHOULD NOT employ the urgent mechanism,
   applications that still decide to employ it MUST set the SO_OOBINLINE
   socket option, such that "urgent data" are delivered inline, as
   intended by the IETF specifications.

[7](#). Security Considerations

   Given that there are two different interpretations of the semantics
   of the Urgent Pointer in current implementations (e.g., depnding on
   the value of the tcp_stdurg sysctl), and that middle-boxes (such as
   packet scrubbers) or the end-systems themselves could cause the
   urgent data to be processed "in band", there exists ambiguity in how
   "urgent data" sent by a TCP will be processed by the intended
   recipient. This might make it difficult for a Network Intrusion
   Detection System (NIDS) to track the application-layer data
   transferred to the destination system, and thus lead to false
   negatives or false positives in the NIDS [CPNI-TCP].

   Probably the best way to avoid the security implications of TCP
   urgent data is to avoid having applications use the TCP urgent
   mechanism altogether. Packet scrubbers could probably be configured
   to clear the URG bit, and set the Urgent Pointer to zero. This would
   basically cause the urgent data to be put "in band". However, this
   might cause interoperability problems or undesired behavior in the
   applications running on top of TCP.

[8](#). IANA Considerations

   This document has no actions for IANA.

[9](#). Acknowledgements

   The authors of this document would like to thank (in alphabetical
   order) David Borman, Wesley Eddy, Alfred Hoenes, Carlos Pignataro,
   Anantha Ramaiah, Joe Touch, Michael Welzl, and Dan Wing for providing
   valuable feedback on earlier versions of this document.

   Additionally, Fernando would like to thank David Borman and Joe Touch
   for a fruitful discussion about TCP urgent mode at IETF 73
   (Minneapolis).

[10](#). References

10.1. Normative References

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

   [RFC1122] Braden, R., "Requirements for Internet Hosts -
              Communication Layers", STD 3, RFC 1122, October 1989.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

   [CPNI-TCP]
              CPNI, "Security Assessment of the Transmission Control
              Protocol (TCP)", (to be published) .

   [Cisco-PIX]
              Cisco PIX, "http://www.cisco.com/en/US/docs/security/asa/
              asa70/command/reference/tz.html#wp1288756".

   [FreeBSD] The FreeBSD project, "http://www.freebsd.org".

   [Linux]   The Linux Project, "http://www.kernel.org".

   [NetBSD]   The NetBSD project, "http://www.netbsd.org".

   [OpenBSD] The OpenBSD project, "http://www.openbsd.org".

   [RFC0961] Reynolds, J. and J. Postel, "Official ARPA-Internet
              protocols", RFC 961, December 1985.

   [UNPv1]   Stevens, W., "UNIX Network Programming, Volume 1.
              Networking APIs: Sockets and XTI", Prentice Hall PTR ,
              1997.

   [Windows2000]
              Microsoft Windows 2000, "http://technet.microsoft.com/
              en-us/library/bb726981(printer).aspx".

   [Windows95]
              Microsoft Windows 95,
              "ftp://ftp.demon.co.uk/pub/mirrors/win95netfaq/
              faq-c.html".

   [phrack]   Ko, Y., Ko, S., and M. Ko, "NIDS Evasion Method named
              "SeolMa"", Phrack Magazine, Volume 0x0b, Issue 0x39, Phile
              #0x03 of 0x12 http://www.phrack.org/

                issues.html?issue=57&id=3#article, 2001.

Appendix A. Survey of the processing of TCP urgent indications by some
             popular TCP implementations

A.1. FreeBSD

   FreeBSD 8.0 [FreeBSD] interprets the semantics of the urgent pointer
   as specified in Section 4 of this document. It does not provide any
   sysctl to override this behavior.

   FreeBSD provides the SO_OOBINLINE socket option that, when set,
   causes TCP "urgent data" to remain "in band". That is, it will be
   accessible by the read(2) call or the recv(2) call without the
   MSG_OOB flag.

   FreeBSD supports only one byte of urgent data. That is, only the
   byte preceding the Urgent Pointer is considered as "urgent data".

A.2. Linux

   Linux 2.6.15-53-386 [Linux] interprets the semantics of the urgent
   pointer as specified in Section 4 of this document. It provides the
   net.ipv4.tcp_stdurg sysctl to override this behavior to interpret the
   Urgent Pointer as specified in RFC 1122 [RFC1122]. However, this
   sysctl only affects the processing of incoming segments (the Urgent
   Pointer in outgoing segments will still be set as specified in
   Section 4 of this document).

   Linux provides the SO_OOBINLINE socket option that, when set, causes
   TCP "urgent data" to remain "in band". That is, it will be
   accessible by the read(2) call or the recv(2) call without the
   MSG_OOB flag.

   Linux supports only one byte of urgent data. That is, only the byte
   preceding the Urgent Pointer is considered as "urgent data".

A.3. NetBSD

   NetBSD 5.0.1 [NetBSD] interprets the semantics of the urgent pointer
   as specified in Section 4 of this document. It does not provide any
   sysctl to override this behavior.

   NetBSD provides the SO_OOBINLINE socket option that, when set, causes
   TCP "urgent data" to remain "in band". That is, they will be
   accessible by the read(2) call or the recv(2) call without the
   MSG_OOB flag.

NetBSD supports only one byte of urgent data. That is, only the byte preceding the Urgent Pointer is considered as "urgent data".

A.4. OpenBSD

OpenBSD 4.2 [OpenBSD] interprets the semantics of the urgent pointer as specified in Section 4 of this document. It does not provide any sysctl to override this behavior.

OpenBSD provides the SO_OOBINLINE socket option that, when set, causes TCP urgent data to remain "in band". That is, they will be accessible by the read(2) or recv(2) calls without the MSG_OOB flag.

OpenBSD supports only one byte of urgent data. That is, only the byte preceding the Urgent Pointer is considered as "urgent data".

A.5. Cisco IOS software

Cisco IOS Software Releases 12.2(18)SXF7, 12.4(15)T7 interpret the semantics of the urgent pointer as specified in Section 4 of this document.

The behaviour is consistent with having the SO_OOBINLINE socket option turned on, i.e. the data is processed "in band".

A.6. Microsoft Windows 2000, Service Pack 4

Microsoft Windows 2000 [Windows2000] interprets the semantics of the urgent pointer as specified in Section 4 of this document. It provides the TcpUseRFC1122UrgentPointer system-wide variable to override this behavior, interpreting the Urgent Pointer as specified in RFC 1122 [RFC1122].

Tests performed with a sample server application compiled using the cygwin environment, has shown that the default behavior is to return the urgent data "in band".

A.7. Microsoft Windows 2008

Microsoft Windows 2008 interprets the semantics of the urgent pointer as specified in Section 4 of this document.

A.8. Microsoft Windows 95

Microsoft Windows 95 interprets the semantics of the urgent pointer as specified in Section 4 of this document. It provides the BSDUrgent system-wide variable to override this behavior, interpreting the Urgent Pointer as specified in RFC 1122 [RFC1122].

Windows 95 supports only one byte of urgent data. That is, only the
byte preceding the Urgent Pointer is considered as "urgent data".
[Windows95]

Appendix B. Changes from previous versions of the draft (to be removed
             by the RFC Editor before publishing this document as an
             RFC)

B.1. Changes from draft-ietf-tcpm-urgent-data-02

   o Addresses WGLC feedback submitted by Michael Welzl, Anantha
      Ramaiah, and Wesley Eddy.

B.2. Changes from draft-ietf-tcpm-urgent-data-01

   o Fixes reference to Cisco IOS Software (layer 8+ stuff ;-)).

   o Cleaned-up Appendix A.5.

B.3. Changes from draft-ietf-tcpm-urgent-data-00

   o Minor editorial changes.

   o Incorporated the specific changes/advice stated in
      http://www.ietf.org/mail-archive/web/tcpm/current/msg04548.html in
      different sections (Section 4, Section 5, Section 6).

B.4. Changes from draft-gont-tcpm-urgent-data-01

   o Draft resubmitted as draft-ietf, as a result of wg consensus on
      adopting the document as a tcpm wg item.

Authors' Addresses

   Fernando Gont
   Universidad Tecnologica Nacional / Facultad Regional Haedo
   Evaristo Carriego 2644
   Haedo, Provincia de Buenos Aires 1706
   Argentina

   Phone: +54 11 4650 8472
   Email: fernando@gont.com.ar
   URI: http://www.gont.com.ar

Andrew Yourtchenko
Cisco
De Kleetlaan, 7
Diegem B-1831
Belgium

Phone: +32 2 704 5494
Email: ayourtch@cisco.com