TCP Maintenance and Minor Extensions (tcpm)
Internet-Draft
Intended status: Standards Track
Expires: August 6, 2009

F. Gont
UTN/FRH
A. Yourtchenko
Cisco

February 2, 2009

On the implementation of TCP urgent data
draft-gont-tcpm-urgent-data-01.txt

Status of this Memo

Copyright Notice

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, and raises awareness about the reliability of TCP urgent indications in the current Internet.

Table of Contents

1. Introduction

TCP incorporates a "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 has been received by the user. This 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].

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].

This document analyzes how 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 the relevant specifications such that they accommodate current practice in processing TCP urgent indications, and also 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 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.

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 TCP urgent data

2.1. Semantics of urgent inications

As discussed in Section 1, 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". 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 TCP urgent mechanism is NOT a mechanism for sending "out-of-band" data: the "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 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 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 a the sysctl "tcp_stdurg" (e.g., net.ivp4.tcp_stdurg) that, when set, supposedly changes the system behavior to interpret the semantics of the TCP Urgent Pointer as described 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.

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].

3.4. Interaction of middle-boxes with urgent data

As a result of the publication of Network Intrusion Detection (NIDs) evasion techniques based on urgent data [phrack] , some middle-boxes modify the TCP data stream such that urgent data is put "in band", that is, they are accessible by the read(2) or recv(2) calls without the MSG_OOB flag. Examples of such middle-boxes are Cisco PIX firewall [Cisco-PIX]. This should discourage applications to depend on urgent data for their corect operation, as urgent data may not be not reliable in the current Internet.

4. Updating RFC 1122

Firstly, 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", we propose that RFC 1122 [RFC1122] be updated 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.

Secondly, we strongly encourage applications that employ Sockets API to set the SO_OOBINLINE socket option, such that urgent data is delivered inline, as intended by the IETF specifications. Furthermore, we discourage the use of the MSG_OOB flag in recv(2) calls to retrieve the "urgent data".

Finally, considering the discussion in Section 3.4, we discourage applications to depend on the TCP urgent mechanism for correct operation, as urgent data may not be reliable in the current Internet.

5. Security Considerations

Given that there are two different interpretations of the semantics of the Urgent Pointer in current implementations, and that either

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 TCP 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 application protocols depend on the use of TCP urgent data 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.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

The authors of this document would like to thank (in alphabetical order) David Borman, Alfred Hoenes, Carlos Pignataro, Anantha Ramaiah, Joe Touch, 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).

8. References

8.1. 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.

8.2. 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 urgent data by some popular implementations

A.1. FreeBSD

FreeBSD [FreeBSD] interprets the semantics of the urgent pointer as specified in RFC 793. It does not provide any sysctl to override this behavior. However, it provides the SO_OOBINLINE that when set

causes TCP urgent data to be put "in band". That is, it will be accessible by the read(2) or recv(2) calls 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 [Linux] interprets the semantics of the urgent pointer as specified in RFC 793. It provides the net.ipv4.tcp_stdurg sysctl to override this behavior to interpret the Urgent Pointer as specified by RFC 1122. However, this sysctl only affects the processing of incoming segments (the Urgent Pointer in outgoing segments will still be set as specified in RFC 793).

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 [NetBSD] interprets the semantics of the urgent pointer as specified in RFC 793. It does not provide any sysctl to override this behavior. However, it provides the SO_OOBINLINE that when set causes TCP urgent data to be put "in band". That is, it will be accessible by the read(2) or recv(2) calls 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 [OpenBSD] interprets the semantics of the urgent pointer as specified in RFC 793. It does not provide any sysctl to override this behavior. However, it provides the SO_OOBINLINE that when set causes TCP urgent data to be put "in band". That is, it 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, versions 12.2(18)SXF7, 12.4(15)T7

Cisco IOS, versions 12.2(18)SXF7, 12.4(15)T7 interpret the semantics of the urgent pointer as specified in RFC 793. However, tests performed with an HTTP server running on Cisco IOS version 12.2(18)SXF7 and 12.4(15)T7 suggest that urgent data is processed "in band". That is, they are accessible together with "normal" data. The TCP debugs on the Cisco IOS device do explicitly mention the

presence of urgent data, and thus we infer that the behavior is different depending on the application.

A.6. Microsoft Windows 2000, Service Pack 4

Microsoft Windows 2000 [Windows2000] interprets the semantics of the urgent pointer as specified in RFC 793. It provides the TcpUseRFC1122UrgentPointer system-wide variable to override this behavior to interpret the Urgent Pointer as specified by RFC 1122. However, the tests performed with the 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 RFC 793.

A.8. Microsoft Windows 95

Microsoft Windows 95 interprets the semantics of the urgent pointer as specified in RFC 793. It provides the BSDUrgent system-wide variable to override this behavior to interpret the Urgent Pointer as specified by RFC 1122. Windows 95 supports only one byte of urgent data. That is, only the byte preceding the Urgent Pointer is considered as "urgent data". [Windows95]

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