EDSA-406
ISA Security Compliance Institute – Embedded Device Security Assurance –

Testing the robustness of implementations of the IETF TCP transport protocol over IPv4 or IPv6

Version 2.01

April 2015

A. DISCLAIMER

ASCI and all related entities, including the International Society of Automation (collectively, “ASCI”)provide all materials, work products and, information (‘SPECIFICATION’) AS IS, WITHOUT WARRANTY AND WITH ALL FAULTS, and hereby disclaim all warranties and conditions, whether express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of reliability or availability, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence, all with regard to the SPECIFICATION, and the provision of or failure to provide support or other services, information, software, and related content through the SPECIFICATION or otherwise arising out of the use of the SPECIFICATION. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION, OR NON-INFRINGEMENT WITH REGARD TO THE SPECIFICATION.

WITHOUT LIMITING THE FOREGOING, ASCI DISCLAIMS ALL LIABILITY FOR HARM TO PERSONS OR PROPERTY, AND USERS OF THIS SPECIFICATION ASSUME ALL RISKS OF SUCH HARM.

IN ISSUING AND MAKING THE SPECIFICATION AVAILABLE, ASCI IS NOT UNDERTAKING TO RENDER PROFESSIONAL OR OTHER SERVICES FOR OR ON BEHALF OF ANY PERSON OR ENTITY, NOR IS ASCI UNDERTAKING TO PERFORM ANY DUTY OWED BY ANY PERSON OR ENTITY TO SOMEONE ELSE. ANYONE USING THIS SPECIFICATION SHOULD RELY ON HIS OR HER OWN INDEPENDENT JUDGMENT OR, AS APPROPRIATE, SEEK THE ADVICE OF A COMPETENT PROFESSIONAL IN DETERMINING THE EXERCISE OF REASONABLE CARE IN ANY GIVEN CIRCUMSTANCES.

B. EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL ASCI OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL,PUNITIVE, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SPECIFICATION, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT OR OTHER SERVICES, INFORMATON, SOFTWARE, AND RELATED CONTENT THROUGH THE SPECIFICATION OR OTHERWISE ARISING OUT OF THE USE OF THE SPECIFICATION, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS SPECIFICATION, EVEN IN THE EVENT OF THE FAULT, TORT (INCLUDING NEGLIGENCE), MISREPRESENTATION, STRICT LIABILITY, BREACH OF CONTRACT OF ASCI OR ANY SUPPLIER, AND EVEN IF ASCI OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Revision history

version	date	changes
1.1	2010.06.15	initial version published to http://www.ISASecure.org
1.41	2010.09.28	added test TCP.T16, probe of unknown application protocols at open TCP ports; create distinct test criteria at high but supported rate and full auto-negotiated link rate; removed protocol conformance aspects of tests since covered by other industry efforts; removed discovery phase since not required to perform uniform testing over all devices; removed mixing of valid and invalid messages in load testing since valid messages create more load on device
2.01	2015.04.13	incorporate EDSA-102 v1.2 errata; change terminology essential services to essential functions; correction of byte 2 WSN figure 3; add sample size to dieharder test TCP.T01; require pseudo random test generation where applicable; remove conformance pass criteria for TCP.T14; load tests run two minutes; clarify pass/fail in Clause 7; lower level protocols must be valid in fuzz tests; incorporate feedback from the tool recognition process in T07, T09, T13; add missing textual list in 4.2.3; added two recently-standardized options and marked others as obsolete (but still could be sent by an attacker); added clarifications that instances of all alternative option classes should be included in testing; technical corrections on TCP.T11

Contents

Foreword

NOTE This is one of a series of robustness test specifications for embedded devices. The full current list of documents related to embedded device security assurance can be found on the web site of the ISA Security Compliance Institute, http:/www.ISASecure.org.

1 Scope

This document is intended to provide requirements for testing the robustness of embedded device server implementations of the IETF TCP protocol, as a measure of the extent to which such implementations defend themselves against

- correctly formed messages and sequences of such messages;
- single erroneous messages; and
- inappropriate sequences of messages;

where failure of the device to continue to provide concurrent automation system control and reporting functions, demonstrates potential security vulnerabilities within the device. This document is not intended to serve as a guide for testing the correctness of implementations or conformance to mandatory provisions of the controlling standard(s), which cannot be achieved solely by observing a device's response to external stimuli.

Requirements are comprised of all the numbered Requirements in this document and any immediately following clarifying information, together with the tables in Clause 7 that describe individual tests. In the event of a conflict between these, the tables in Clause 7 take precedence.

NOTE 1 The TCP protocol is stateful, with both server and client roles. Embedded devices usually implement primarily the server role of TCP, though they may themselves need the client role for some management coordination functions.

NOTE 2 Although conformance is explicitly NOT a goal of this testing, prior versions of this document included some aspects of conformance testing which have now intentionally been removed.

2 Normative references

This associated specification contains requirements common to this and similar robustness tests for other protocols for embedded devices, including requirements on test configurations.

[EDSA-310] *ISA Security Compliance Institute – Embedded device security assurance – Requirements for embedded device robustness testing* [1], as specified at http://www.ISASecure.org

NOTE 1 Within this document, references to specific subclauses of this normative reference are made through symbolic tags of the form [CRT.Symbolic_tag]; the resolution of those tags is made in [EDSA-310], Table 1.

These publications of the Internet Engineering Task Force (IETF) are the controlling specifications for the protocol whose robustness testing is the subject of this document:

NOTE 2 For each RFC*nnn*, the controlling version can be found at http://tools.ietf.org/html/rfc*nnn*.

[Port_numbers] *IANA port numbers*, as specified at http://www.iana.org/assignments/port-numbers

RFC793, *Transmission control protocol*

RFC1122, *Requirements for internet hosts – communication layers*

NOTE 3 Only 4.2 is referenced.

RFC1146, *TCP alternate checksum options*

RFC1191, *Path MTU discovery*

NOTE 4 Only 3.1 and 6.4 are referenced.

RFC1323, *TCP extensions for high performance*

[1] to be published concurrently with this document

RFC1644, *T/TCP -- TCP extensions for transactions - functional specification*

RFC1693, *An extension to TCP: Partial order service*

RFC1812, *Requirements for IP version 4 routers*

NOTE 5 Only 6.2 is referenced, and then only for DUTs that also act as IP routers.

RFC2018, *TCP selective acknowledgment options*

RFC2385, *Protection of BGP sessions via the TCP MD5 signature option*

RFC2883, *An extension to the selective acknowledgement (SACK) option for TCP*

RFC2988, *Computing TCP's retransmission timer*

RFC3168, *The addition of explicit congestion notification (ECN) to IP*

NOTE 6 Only 6.1 is referenced, which adds the CWR and ECE flags and their procedure to TCP.

RFC3390, *Increasing TCP's initial window*

RFC5681, *TCP congestion control*

NOTE 7 Other IETF specifications related to the above can be found in the Bibliography.

RFC5482, *TCP User Timeout Option*

RFC5925, *The TCP Authentication Option*

RFC6247, *Moving the Undeployed TCP Extensions RFC 1072, RFC 1106, RFC 1110, RFC 1145, RFC 1146, RFC 1379, RFC 1644, and RFC 1693 to Historic Status*

These publications of the Internet Engineering Task Force (IETF) are the controlling specifications for two higher-layer protocols that can be used for testing the robustness of the TCP protocol that is the subject of this document.

RFC862, *Echo protocol*

3 Definitions and abbreviations

3.1 Definitions

3.1.1
CC*
any of the TCP options CC, CCNEW or CCECHO

3.1.2
device under test
device that is being stimulated and observed during testing to demonstrate the characteristics and behavior of the device when presented with the selected sequence of test stimuli

3.1.3
erroneous (message or PDU or option)
PDU that violates either syntactic rules on PDU structure or semantic rules on PDU content or both, or PDU option that violates either syntactic rules on PDU option structure or semantic rules on PDU option content or both

NOTE 1 Semantic and syntactic rule violations can interact, as when the value of one field determines the size of another field.

NOTE 2 The term erroneous includes syntactic malformation, semantically invalid values, and contextually invalid values and sequences

NOTE 3 This is addressed further in [CRT.Terminology_of_Erroneous].

3.1.4
“Ethernet”
either the IETF Ethernet II protocol or IEEE 802 SNAP over IEEE 802.2 Type 1 LLC over IEEE 802.3

3.1.5
fragmenting
function performed by IPv4 to map one unfragmented NPDU into multiple smaller fragmented NPDUs before transmission

NOTE The equivalent OSI terms is segmenting, as specified in ISO/IEC 7498 1:1994, 5.8.1.9.

3.1.6
inferior (protocol)
protocol at a lower layer or sublayer than the referenced protocol

3.1.7
lower tester
tester that controls and observes a protocol layer implementation in a DUT through stimulus and observation via lower protocol layers and a physical interconnection to the TD

NOTE This is the only type of testing used in the ISCI EDSA robustness tests.

3.1.8
malformed (message or PDU)
PDU that violates syntactic rules on PDU structure

NOTE This is addressed further in [CRT.Terminology_of_Erroneous].

3.1.9
reassembling
post-reception function performed by IP to reconstruct one unfragmented NPDU from multiple fragmented NPDUs

3.1.10
superior (protocol)
protocol at a higher layer or sublayer than the referenced protocol

3.1.11
testing device
conceptual single network-connected device, possibly consisting of multiple physical network-connected devices, used to test the robustness of the device under test

NOTE This could be any programmable network-connected device capable of processing PDUs at the rate required for testing.

3.1.12
upper tester
tester that controls and observes a protocol layer implementation in a DUT through stimulus and observation via a DUT-internal service interface between test software and the protocol layer under test

3.1.13
vulnerability
flaw or weakness in a system's design, implementation, operation, or management that could be exploited to violate the system’s integrity or security policy

3.2 Abbreviations

The following abbreviations are used in this document

ACK	[TCP flag] acknowledgment field is significant
APDU	application-layer protocol data unit

CRT	communication robustness testing
CWR	[TCP flag] congestion window reduced
DPDU	data-link-layer protocol data unit
DUT	device under test
ECE	[TCP flag] ECN echo
ECN	explicit congestion notification
FIN	[TCP flag] finish – sender has concluded sending data
FSM	finite state machine
GPL	gnu public license
IANA	Internet assigned numbers authority
ICMP	Internet control message protocol
IETF	Internet engineering task force
IP	Internet protocol (IETF network layer protocol)
IPv4	IP version 4 (uses 32-bit network layer addresses)
IPv6	IP version 6 (uses 128-bit network layer addresses)
NIST	(U.S.) National Institute of Standards and Technology
(*N*)PDU	(*N*-layer) protocol data unit, where *N* = D (data-link), N (network), T (transport), A (application), etc
NPDU	network-layer protocol data unit
PSH	[TCP flag] push function, causes queued fragments to be transmitted as soon as possible
RFU	reserved for future use
RST	[TCP flag] reset the connection
STS	statistical test suite
SYN	[TCP flag] synchronize sequence numbers
TCP	transmission control protocol
TD	testing device
TPDU	transport-layer protocol data unit
URG	[TCP flag] urgent pointer field is meaningful

4 Elements of the protocol under test

4.1 General

This document specifies robustness testing for the IETF TCP protocol, which is a stateful transport protocol providing an ordered, prioritizable, reliable end-to-end communications channel.

4.2 TCP TPDUs

4.2.1 TCP TPDU structure

A TCP TPDU is structured as shown in Figure 1, using a big-endian octet order.

```
         0                   1                   2                   3
         0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
-------- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |          SourcePort           |        DestinationPort        |
   H     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                        SequenceNumber                         |
   E     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                    AcknowledgmentNumber                       |
   A     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |  Data |       |C|E|U|A|P|R|S|F|                               |
   D     | Offset|  RFU  |W|C|R|C|S|S|Y|I|            Window             |
         |       |       |R|E|G|K|H|T|N|N|                               |
   E     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |           Checksum            |         UrgentPointer         |
   R     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |               varying-length Options (if any)                 |
         |      (1B granularity, padded with 0x00 to a 4B multiple)      |
-------- +-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-+
D A T A  |              /  varying-length Data (if any) /                |
         |              /   (1B granularity, unpadded)  /                |
-------- +-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-+
```

Figure 1 – TCP TPDU structure

4.2.2 Mandatory fields

The following fields are mandatory components of each TCP TPDU (where field sizes are specified in octets (B) or bits (b)):

a) SourcePort (SP): (2B): source TCP-TSAP-identifier; default 0x0000. See [Port_numbers]

b) DestinationPort (DP): (2B): destination TCP-TSAP-identifier. See [Port_numbers]

c) SequenceNumber (SN): (4B): If SYN is reset, this field specifies the sequence number of the first data octet in this PDU. If SYN is set, the sequence number (SN) is the initial sequence number (ISN) and the first data octet is ISN+1.

d) AcknowledgmentNumber (AN): (4B): If ACK is set, this field specifies the next sequence number that the sender of the PDU is expecting to receive. ACK is always set once a connection is established.

e) DataOffset (DO): (4b): Size of TCP header in 4B words, range 5..15

f) RFU: (4b): coded as zero on transmission; ignored on reception

g) TCP flags: (8b):

   — CWR (1b): 1=Congestion window reduced (per RFC3168, 6.1)

   — ECE (1b): 1=ECN echo (per RFC3168, 6.1)

   — URG (1b): 1=Urgent pointer field is relevant

   — ACK (1b): 1=Acknowledgment field is relevant

   — PSH (1b): 1=Push function, causes queued fragments to be transmitted ASAP

   — RST (1b): 1=Reset the connection

   — SYN (1b): 1=Synchronize sequence numbers

   — FIN (1b): 1=Sender has concluded sending data

h) Window (W): (2B): The number of data octets beginning with the one indicated in the acknowledgment field (AF) that the sender of this PDU is willing to accept.

   NOTE Thus the sender's current acceptance window is data octets [AN .. AN+W-1] inclusive.

i) Checksum: (2B): unkeyed message integrity code. See 4.2.4.2

j) UrgentPointer (UP): (2B): If URG is set, this field specifies the offset relative to SN of the last urgent data octet in this TPDU (per RFC1122, 4.2.2.4). In other words, when URG is set, urgent data is present in the TPDU and this field specifies the zero-origin size of the urgent data.

   NOTE The unstated presumption of the RFCs is that the urgent data occupies the first (UP+1) payload octets of the TPDU, octets [SN .. SN+UP] inclusive, with the normal data occupying any remaining TPDU octets.

k) Options: (0B .. 40B length, granularity 1B, padded with bytes of 0x00 to a 4B boundary)

l) Data: (varying length ≥ 0B, granularity 1B, not padded to a multi-octet boundary)

4.2.3 Optional fields

TCP defines two types of option, each structured generically as shown in Figure 2, using a big-endian octet order:

```
                    +------+
single-octet option | Op # |
                    +------+
                    |  1B  |

                    +------+----------------------------+--------------+
 multi-octet option | Op # | RemainingOptionLength (ROL) | RemainingData|
                    +------+----------------------------+--------------+
                    |  1B  |             1B              |    <ROL> B   |
```

Figure 2 – Generic option structure

Seventeen specific options are defined for TCP, as shown in **Error! Reference source not found.** and Figure 4.

- EOS (1B): end-of-option-sequence, used to pad the Options field to a 4B multiple;
- PAD (1B): intra-option padding used to force a specific starting alignment of a subsequent option;
- MSSN (4B): maximum receive segment size for the sender, which is valid only when sent in the initial connection request (e.g., when the SYN flag is set).
- WSN (3B): window scaling
- SACKN (2B): enable selective acknowledgements to specifying peer
- SACK (2+8*n* B): convey *n* selective acknowledgment intervals to peer

  The following two options are deprecated, having been replaced by the TS option.
- ECHORQ (3B): request to peer to echo the specified option byte, thereby inducing peer transmission.
- ECHORP (3B): peer reply to receipt of an ECHORQ option.
- TS (10B): time sync
- POCR (2B): Partial order connection permitted
- POCSP (3B): Partial order connection service profile
- CC (4B): connection count, to support transaction-TCP
- CC.NEW (4B): same as CC, plus notifies peer to flush CC cache
- CC.ECHO (4B): echoes connection count of an initial SYN in a SYN+ACK TPDU
- ACR (3B): alternate checksum request, can select an 8-bit or 16-bit Fletcher checksum
- ACD (2+n B): alternate checksum data – conveys additional checksum data beyond the standard 3B
- MD5 (18B): uses an MD5 signature in lieu of a checksum

NOTE: The status of TCP options is summarized at https://www.ietf.org/mail-archive/web/tcpm/current/msg03199.html. TCP options not considered obsolete are found in RFC 6247

```
            +-------+
     EOS    |   0   |
            +-------+
            |  1B   |

            +-------+
     PAD    |   1   |
            +-------+
            |  1B   |

            +-------+--------+----------------+
      MSSN  | Type  | Length | MaxSegmentSize |
            | (=2)  |  (=4)  |                |
  (RFC793)   +-------+--------+----------------+
            |  1B   |   1B   |       2B       |

            +-------+--------+--------------+
      ECHORQ| Type  | Length | ConveyedData |
            | (=6)  |  (=6)  |              |
 (RFC1072)  +-------+--------+--------------+
(deprecated)|  1B   |   1B   |      1B      |

            +-------+--------+--------------+
      ECHORP| Type  | Length | ConveyedData |
            | (=7)  |  (=6)  |              |
 (RFC1072)  +-------+--------+--------------+
(deprecated)|  1B   |   1B   |      1B      |

            +-------+--------+----------+---------------+
      TS    | Type  | Length | TS value | TS echo reply |
            | (=8)  |   10   |          |               |
 (RFC1323)  +-------+--------+----------+---------------+
            |  1B   |   1B   |    4B    |      4B       +

            +-------+--------+------------+
      WSN   | Type  | Length | ShiftCount |
            | (=3)  |  (=3)  |            |
 (RFC1323)  +-------+--------+------------+
            |  1B   |   1B   |     1B     |

            +-------+--------+
      SACKN | Type  | Length |
            | (=4)  |  (=2)  |
 (RFC2018)  +-------+--------+
            |  1B   |   1B   |

            +-------+--------+--------------+-------------+---//----+
      SACK  | Type  | Length | Left Edge 1  | Right Edge 1|         |
            | (=5)  |  2+4n  |  (repeating substructure)  |  ...    |
 (RFC2018)  +-------+--------+--------------+-------------+---//----+
            |  1B   |   1B   |      2B      |     2B      + 4(n-1)B +

            +-------+--------+
      POCR  | Type  | Length |
            | (=9)  |  (=2)  |
 (RFC1693)  +-------+--------+
 (obsolete) |  1B   |   1B   |
```

Figure 3 – Specific option structure (1/2)

```
     POCSP | Type   | Length | Flags |
           | (=10)  |  (=3)  |       |
(RFC1693)  +-------+--------+-------+
(obsolete) |  1B   |   1B   |   1B  |

           +-------+--------+-----------------+
     CC    | Type   | Length | ConnectionCount |
           | (=11)  |  (=6)  |                 |
(RFC1644)  +-------+--------+-----------------+
(obsolete) |  1B   |   1B   |        4B       |

           +-------+--------+-----------------+
     CCNEW | Type   | Length | ConnectionCount |
           | (=12)  |  (=6)  |                 |
(RFC1644)  +-------+--------+-----------------+
(obsolete) |  1B   |   1B   |        4B       |

           +-------+--------+-----------------+
     CCECHO| Type   | Length | ConnectionCount |
           | (=13)  |  (=6)  |                 |
(RFC1644)  +-------+--------+-----------------+
(obsolete) |  1B   |   1B   |        4B       |

           +-------+--------+--------------+
     ACR   | Type   | Length | checksumType |
           | (=14)  |  (=3)  |              |
(RFC1146)  +-------+--------+--------------+
(obsolete) |  1B   |   1B   |      1B      |

           +-------+--------+----------+
     ACD   | Type   | Length | checksum |
           | (=15)  | (=2+n) |          |
(RFC1146)  +-------+--------+----------+
(obsolete) |  1B   |   1B   |    nB    |

           +-------+--------+--------+
     MD5   | Type   | Length | Digest |
           | (=19)  |  (=16) |        |
(RFC2385)  +-------+--------+--------+
(obsolete) |  1B   |   1B   |   14B  |

           +-------+--------+-----------------+
     UTO   | Type   | Length | TimeoutInterval |
           | (=28)  |    4   |                 |
(RFC5482)  +-------+--------+-----------------+
           |  1B   |   1B   |       2B        +

           +-------+--------+--------+-----------++--------+
    TCP-AO | Type   | Length | KeyID | RNextKeyID |    MAC  |
           | (=29)  |   4+n  |       |            |         |
(RFC5925)  +-------+--------+-------+------------+---------+
           |  1B   |   1B   |  1B   |     1B     |    nB   |
```

Figure 4 – Specific option structure (2/2)

The MSSN option is only meaningful when the SYN flag is set in the same TCP TPDU. It is required that both sender and receiver support this option, and that this option be used when the desired MSSN value differs from the default of 536 B (per RFC1122, 4.2.2.6 and RFC1191, 3.1).

4.2.4 Mandatory protocol aspects

4.2.4.1 Conveying IP NPDU

TCP fixes the value of one of the header fields of any conveying IP NPDU.

- For IPv4, the NPDU header SHALL specify the TCP protocol in its ProtocolType field:
  - ProtocolType: 0x06 (TCP)
- For IPv6, the last header in the NPDU SHALL specify the TCP protocol in its NextHeader field:
  - NextHeader: 0x06 (TCP)

TCP specifies that some options of the conveying IP NPDU, such as Source Route, are to be passed to TCP, which is then to use the reverse route from that Source Route option for all future transmissions to the remote peer (per RFC1122, 4.2.3.8).

4.2.4.2 Checksum procedure

The checksum field is the 16-bit one's complement of the one's-complement sum of all 2 B words, in big-endian octet order, in a virtual TCP TPDU created by prefixing the actual TPDU by a pseudo-header consisting of extra 4 B words. The pseudo-header for use with IPv4 is shown in Figure 5; the pseudo-header for use with IPv6 is shown in Figure 6.

Each pseudo-header contains the Source IP Address; the Destination IP Address; the IP protocol ID for TCP (0x06), which in IPv6 is used as the value of the NextHeader field of the last header in the IPv6 NPDU; and the length in octets of the conveyed TCP TPDU. Together these give the TCP TPDU some protection against being reconstructed from misrouted fragmented NPDUs.

NOTE This pseudo-header information is carried in the IP NPDU and is transferred across the Transport/Network layer service interface.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Source IPv4 Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Destination IPv4 Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x00      |     0x06      |     TCP TPDU length per NL    |
+-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-+
|              varying-length TCP TPDU of Figure 1              |
|              (padded with zero to a 2B boundary)              |
+-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-+
```

Figure 5 – Virtual TCP over IPv4 TPDU used for checksum computation

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                                                             -+
|                                                               |
+-                     Source IPv6 Address                     -+
|                                                               |
+-                                                             -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                                                             -+
|                                                               |
+-                   Destination IPv6 Address                  -+
|                                                               |
+-                                                             -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     TCP TPDU length per NL                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    0x000000                   |      0x06     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              varying-length TCP TPDU of Figure 1              |
|              (padded with zero to a 2B boundary)              |
+-+-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+-/-+-+-+-+-+-+-+
```

Figure 6 – Virtual TCP over IPv6 TPDU used for checksum computation

If a TCP TPDU contains an odd number of octets to be checksummed, an octet of zeros is appended when forming the virtual TCP TPDU that is used for checksum computation, to ensure an integral number of 2B words, as shown in Figure 5 and Figure 6. Within the virtual TCP TPDU, while computing the checksum, the checksum field of the contained varying-length TCP TPDU is set to zero.

The TCP TPDU length in the third 4B word of the virtual header is the length of the TPDU as presented to or reported by the associated network layer. This is typically the TCP header length plus the computed data length in octets, not counting the extra prefix or padding suffix octets of the virtual TCP TPDU.

4.2.4.3 TCP state machine

The state machine for TCP is shown in Figure 7, with transitions as listed in Table 1.

NOTE 1 This state machine includes all amendments from RFCs updating TCP.

NOTE 2 An alternative more-readable state machine can be found at:
http://www4.informatik.uni-erlangen.de/Projects/JX/Projects/TCP/tcpstate.gif

```
    ________      g        ________
 h |        |<------------|        |
-->| CLOSED |------------>| LISTEN |
   |________|       ------|________|
      |  ^         /        ^   |
      |  -------  /         |   |
      |         \/          |   |
     a|       a'/\        b'|  b|
      |        /  \T'       |   |
      |       /    \        |   |
      |      /      \    c' |   |
      |     /  -------------|---|----------|
      |    /  /       \     |   |          |
    __V___V_ /         \ ___|___V        __V_____                    ________
   | SYN-   |     b     |  SYN-  |  c   |ESTAB-  |  e               | CLOSE- |
   |   SENT |---------->|RECEIVED|----->|  LISHED|----------------->|   WAIT |
   |________|           |________|      |________|                  |________|
                          |                |                            |
                         d|               d|                           d|
                          |                |                            |
                          |          ______V_          ____V___      ___V____
                          |         |  FIN-  |    e   |        |    | LAST-  |
                          --------->| WAIT-1 |------->|CLOSING |    |   ACK  |
                                    |________|        |________|    |________|
                                         |   \            |             |
                                         |    \  e'       |             |
                                        f|     ------    f|           f'|
                                         |           \    |             |
                                      ___V____        V___V___       ___V____
                                     |  FIN-  |   e  |TIME-   |  T  |        |
                                     | WAIT-2 |----->|   WAIT |---->| CLOSED |
                                     |________|      |________|     |________|
```

Figure 7 – TCP state machine

Table 1 – TCP state machine transitions

Label	Event	Action
a	active application OPEN	create TCB, send SYN
a'	active application OPEN	send SYN
b	receive SYN	send SYN,ACK(SYN)
b’	receive RST	—
c	receive ACK(SYN)	—
c'	receive SYN,ACK(SYN)	send ACK
d	application CLOSE	send FIN
e	receive FIN	send ACK(FIN)
e'	receive FIN,ACK	send ACK(FIN)
f	receive ACK(FIN)	—
f'	receive ACK(FIN)	delete TCB
g	application CLOSE	delete TCB
h	passive application OPEN	create TCB
T	timeout=2MSL	delete TCB
T’	timeout=2MSL	send RST, delete TCB

The full state machine for the transactional version of TCP described in RFC1644 is shown in Figure 8, with the corresponding transitions shown in Table 2. As noted in the footnote to Figure 8 and its state transition table, the T/TCP state machine is really just an augmented version of Figure 7 and Table 1, where that augmentation consists of adding a single Boolean variable to the state to track whether a SYN or FIN (depending on the base state) still needs to be sent to the TCP peer.

```
  ________      g        ________
 |        |<------------|        |
 | CLOSED |------------>| LISTEN |
 |________|  h    ------|________|
      |          /        |     |
      |         /        i|    j|
      |        /          |     |
     a|     a'/           |    _V______              ________
      |      /     j      |   |ESTAB-  |       e'    | CLOSE- |
      |     /  -----------|-->| LISHED*|------------>|   WAIT*|
      |    /  /           |   |________|             |________|
      |   /  /            |    |     |                |     |
      |  /  /             |    |    c|              d'|    c|
  ____V_V_ /       _______V    |   __V_____           |   __V_____
 | SYN-   |   b'  |  SYN-  |c  |  |ESTAB-  |  e       |  | CLOSE- |
 |   SENT |------>|RECEIVED|---|->|  LISHED|----------|->|   WAIT |
 |________|       |________|   |  |________|          |  |________|
    |              |          |     |                |        |
    |              |          |     |              __V_____   |
    |              |          |     |             | LAST-  |  |
  d'|            d'|        d'|    d|             |  ACK*  |  |
    |              |          |     |             |________|  |
    |              |          |     |                     |   |
    |              |   ______V_     |       ________      |c'  |d
    |          k   |   |  FIN-  |   |  e''' |        |     |   |
    |        -------|-->| WAIT-1*|---|------>|CLOSING*|    |   |
    |       /       |   |________|   |       |________|    |   |
    |      /        |          |     |            |        |   |
    |     /         |        c'|     |          c'|        |   |
 ___V___ /      ____V___      V_____V_      ____V___    V___V___
| SYN-   | b'' |  SYN-  |  c  |  FIN-  | e'' |        |   | LAST-  |
|  SENT* |---->|RECEIVD*|---->| WAIT-1 |---->|CLOSING |   |   ACK  |
|________|     |________|     |________|     |________|   |________|
                                    |               |           |
                                   f|              f|         f'|
                                 ___V____       ____V___     ___V____
                                |  FIN-  | e   |TIME-   | T |        |
                                | WAIT-2 |---->|   WAIT |-->| CLOSED |
                                |________|     |________|   |________|
```

LEGEND

SYN-SENT* and SYN-RECEIVED* (shown in the figure at SYN-RECEIVD*) differ from the SYN-SENT and SYN-RECEIVED state, respectively, in recording the fact that a FIN needs to be sent. The other starred states – ESTABLISHED*, LAST-ACK*, FIN-WAIT-1*, CLOSING* – indicate that the connection is half-synchronized (i.e., that a SYN needs to be sent).

NOTE This FSM can be replaced by that of Figure 7 through addition of a *starredState* auxiliary Boolean variable to that FSM, with additional same-state transitions where required to go from starred to unstarred status.

Figure 8 – T/TCP state machine

Table 2 – T/TCP state machine transitions

Label	Event	Action
a	active OPEN	create TCB, send SYN
a'	active OPEN	send SYN
b	receive SYN [no TAO]	send ACK(SYN)
b'	receive SYN [no TAO]	send SYN,ACK(SYN)
b''	receive SYN [no TAO]	send SYN,FIN,ACK(SYN)
c	receive ACK(SYN)	—
c'	receive ACK(SYN)	send FIN
d	CLOSE	send FIN
d'	CLOSE	send SYN,FIN
e	receive FIN	send ACK(FIN)
e'	receive FIN	send SYN,ACK(FIN)
e''	receive FIN	send FIN,ACK(FIN)
e'''	receive FIN	send SYN,FIN,ACK(FIN)
f	receive ACK(FIN)	—
f'	receive ACK(FIN)	delete TCB
g	CLOSE	delete TCB
h	passive OPEN	create TCB
i (= b')	receive SYN [no TAO]	send SYN,ACK(SYN)
j	receive SYN [TAO OK]	send SYN,ACK(SYN)
k	receive SYN [TAO OK]	send SYN,FIN,ACK(SYN)
T	timeout=2MSL	delete TCB

4.2.4.4 Sequencing concepts

4.2.4.4.1 General

Sequence numbers are unsigned 32-bit values, for which all arithmetic and comparison operations are modulo 2^{32}. Concepts of “less than” and “greater than” are with respect to the arithmetic difference of two unsigned 32-bit values, when that difference is interpreted as a signed two’s-complement 32-bit value:

- “less than” means that the difference has an apparent negative value;
- “greater than” means that the difference has an apparent non-zero positive value.

Table 3 lists the conceptual state variables associated with a TCP connection, and the associated variables conveyed as implicit or explicit parameters of a TCP TPDU.

NOTE The structured names of most of these variables are identical to those specified in RFC793. However, those variable names in RFC793 which were inconsistent with the rest have been renamed to follow the SND and RCV and SEQ component structuring of the other names.

Table 3 – TCP connection: Conceptual state and TPDU variables

Variable name	Use
Send sequence variables	
SND.RRSS	remote receive segment size (negotiated option)
SND.UNA	send unacknowledged
SND.NXT	send next
SND.WND	send window
SND.UP	send urgent pointer
SND.WUSN	segment sequence number used for last window update
SND.WUAN	segment acknowledgment number used for last window update
SND.ISN	initial send sequence number
Receive sequence variables	
RCV.RSS	receive segment size (negotiated option)
RCV.NXT	receive next
RCV.WND	receive window
RCV.UP	receive urgent pointer
RCV.ISN	initial receive sequence number
Current segment variables	
SEG.SEQ	sequence number of first octet of segment
SEG.ACK	segment acknowledgment number – next SEG.SEQ expected by receiver
SEG.LEN	segment length
SEG.WND	segment window
SEG.UP	segment urgent pointer
SEG.PRC	segment precedence value

4.2.4.4.2 TCP connection sequencing concepts and constraints

NOTE These constraints are from RFC793.

The payload octets of a TCP TPDU convey both APDU data for the application protocol associated with the specified ports of the TCP connection. When present, the SYN and FIN flags each occupy one octet of the TCP connection’s sequence number space, but without being represented by distinct octets of the TPDU. The non-present SYN octet is assigned the first sequence number of the connection, with the first actual TPDU payload octet assigned the next sequence number. Similarly, the non-present FIN octet is assigned the last sequence number of the connection, after the sequence number of the last octet of actual TPDU payload, if any. Thus

SEG.LEN = SIZEOF(TPDU) – (TPDU.DataOffset $\times$ 4) + TPDU.SYN + TPDU.FIN , where the latter two flags are considered 1-bit unsigned integer values

SEG.SEQ + TPDU.SYN (modulo 2^{32}) = sequence number of first payload octet, if any, of a segment

SEG.SEQ + SEG.LEN − TPDU.FIN − 1 (modulo 2^{32}) = sequence number last payload octet, if any, of a segment

SEG.SEQ + SEG.LEN − 1 (modulo 2^{32}) = last sequence number of a segment

A new acknowledgment (called an "acceptable ack"), is one for which the inequality below holds:

SND.UNA $<$ SEG.ACK $\le$ SND.NXT (modulo 2^{32})

A segment on the retransmission queue is fully acknowledged if the sum of its sequence number and length is less than or equal to the acknowledgment value in the received segment.

A segment is judged to occupy a portion of valid receive sequence space if

$\text{RCV.NXT} \leq \text{SEG.SEQ} < \text{RCV.NXT} + \text{RCV.WND}$ (modulo 2^{32})

or

$\text{RCV.NXT} \leq \text{SEG.SEQ}+\text{SEG.LEN} - 1 < \text{RCV.NXT} + \text{RCV.WND}$ (modulo 2^{32})

Actually, due to zero window sizes and zero length segments, there are four cases for the acceptability of a received segment, as enumerated in RFC793 and Table 4:

Table 4 – TCP connection: Acceptability of a received segment

Segment receive test		
Length	**Window**	**Action (modulo 2^{32})**
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT ≤ SEG.SEQ < RCV.NXT + RCV.WND
>0	0	not acceptable
>0	>0	RCV.NXT ≤ SEG.SEQ < RCV.NXT + RCV.WND or RCV.NXT < SEG.SEQ + SEG.LEN ≤ RCV.NXT + RCV.WND
NOTE The above set of cases from RFC793 exclude the case		
>0	>0	SEG.SEQ < RCV.NXT and RCV.NXT + RCV.WND < SEG.SEQ + SEG.LEN
where the received segment spans and extends beyond both sides of the receive window		

When the receive window is zero, only segments containing a set ACK flag SHALL be accepted. Thus, it is possible for a TCP connection to maintain a zero receive window while transmitting data and receiving ACKs. However, even when the receive window is zero, a TCP connection SHALL continue to process the RST, URG and FIN fields of all received segments.

The sequence-numbering scheme for TPDU payload octets also is used to protect certain control information, by implicitly including some control flags (when set) in the sequence space so that they can be retransmitted and acknowledged without confusion (i.e., one and only one copy of the control will be acted upon). Control information is not carried physically as segment data, but it occupies sequence space. SYN and FIN are the only controls requiring this protection, and these controls are intended for use only at connection opening and closing. For sequence number purposes, SYN is considered to occur as a virtual control octet immediately before the first actual data octet of the segment in which the SYN flag is set, while FIN is considered to occur as a virtual control octet immediately after the last actual data octet of the segment in which the FIN flag is set. In either case, the resulting segment length (SEG.LEN) includes both data and control sequence space. When a SYN is present then SEG.SEQ is the sequence number of the SYN, not of the first (if present) data octet of the segment.

4.2.5 Elements of procedure: Connection establishment via 3-way handshake

TCP connections are established via a 3-way protocol “handshake”. Classic attacks on TCP servers attempt to get the server to allocate state variables when responding to the first phase of the “handshake”, receipt of a TPDU with the SYN flag set while the server is in a LISTEN state. Techniques have been developed to encode enough of the server state in the initial sequence number returned by the server in the TPDU that represents the second phase of the “handshake”, which is expected to be returned by the requesting client during the third phase of the “handshake”; this permits the server to defer allocation of resources to the connection until after receipt of the TPDU representing that final third of the “handshake”. To avoid spoofing, this information SHOULD be encoded via an algorithm resistant to attacker analysis and prediction.

An embedded device implementation of TCP server functionality SHOULD include such protective measures; any implementation that does not is trivially vulnerable to resource depletion attacks, where an attacker initiates many connections but leaves them “half-open” by never completing the third phase of the “handshakes”.

4.2.6 Optional TPDU components and elements of procedure

4.2.6.1 General

The only optional TPDU components are the options of the TCP option field.

4.2.6.2 Receiver maximum segment size

RFC793 specifies a mechanism for increasing or decreasing the default receiver segment size.

The MSSN option MAY be notified only when the TCP TPDU’s SYN flag is set, and SHOULD NOT be present when the SYN flag is reset. Receiver support of this option is mandatory.

4.2.6.3 TCP echo (deprecated for timestamp use)

RFC1072 specifies an optional mechanism for sending a short 4B string such as a TCP timestamp on a TCP connection. When used as a timestamp, this mechanism was replaced by that of 4.2.6.4, so SHOULD NOT be employed by software designed after 1995. However, this mechanism can echo more than timestamps; for any other echo purpose its use is not deprecated.

4.2.6.4 Timestamp

RFC1323 specifies an optional low-overhead mechanism for providing a measurement of current round-trip communications delay on a TCP connection, for use in dynamic adjustment of retransmission timers. Any sender-originated timestamp value SHALL be the low-order octets of a monotonic clock of the sender’s that progresses approximately linearly with time.

4.2.6.5 Window scaling

RFC1323 specifies an optional mechanism for increasing the number of bytes that a TCP connection may have in transit, unacknowledged. The TCP option WSN notifies the TCP peer

- that the sending peer supports window scaling, and
- of a non-negative binary scale factor by which the receiving peer should multiply the receive window edge value(s) found in a TCP header to obtain the true receive window boundaries.

After notification from the TCP peer, a receiving TCP entity augments its receive state variables of Table 3 with an additional scaling state variable. Similarly, the notifying peer augments its sending state variables with a corresponding scaling state variable.

The WSN option MAY be notified only when the TCP TPDU’s SYN flag is set, and SHOULD NOT be present when the SYN flag is reset. Notification of option use is unilateral, not requiring the peer TCP entity's consent.

4.2.6.6 Selective acknowledgment

RFC2018 specifies an optional mechanism for acknowledging segments of the TCP transmission stream while earlier-occurring segments remain unacknowledged, potentially requiring retransmission. After notification from the TCP peer (via a SACKN option), a sending TCP entity augments the state variables of Table 3 with additional information to track which non-continuous segments, if any, of the receive window have been acknowledged.

The SACKN option MAY be notified only when the TCP TPDU’s SYN flag is set, and SHOULD NOT be present when the SYN flag is reset. Once notified, the SACK option SHOULD be included by the acknowledging TCP peer in all ACK TPDUs which do not acknowledge the highest sequence number in the acknowledger’s receive queue. If a TCP peer has not received a SACKN option from its remote peer in a TPDU with the SYN flag set, it SHALL NOT send TPDUs containing the SACK option.

Notification of option use is unilateral, not requiring the peer TCP entity's consent.

4.2.6.7 Partially ordered connection (experimental, obsolete)

RFC1693 specifies an experimental mechanism for delivering segments of a TCP transmission stream in a different order than the originated by the peer TCP entity, After bilateral negotiation via TPDUs with the SYN flag set, containing the POCR request/reply option, either TCP peer MAY augment some TPDUs with begin-profile and end-profile markers (via the POCSP option), indicating that the infix of the TCP transmission stream whose first and last segment(s) are so marked contains a partially-order-connection service profile.

Negotiation of option use is bilateral, requiring the peer TCP entity's consent via a returned POCR option.

NOTE The details of the conveyed partial order service profile(s) appear to be unspecified in the RFC. Hence it is unlikely that any DUT will implement this option.

4.2.6.8 Transaction support (experimental, obsolete)

RFC1644 specifies an experimental mechanism for supporting a series of rapidly established and released TCP connections between the same pair of peer TCP entities. The basic TCP FSM of Figure 7 can be retained by the addition of a *starredState* Boolean variable to the FSM, effectively embodying the FSM of Figure 8 and the extended state transition table of Table 2.

The TCP options CC and CCNEW notify a receiving TCP entity that the sending entity wishes to start or continue a sequence of rapidly opened and closed TCP connections. If the receiving TCP peer agrees and has cached state from a recent such connection with the corresponding TCP entity, it includes a CC or CCECHO option in its reply and makes the accelerated transitions in its FSM; otherwise it behaves as if the CC/CCNEW had not been received.

The transaction support option (known as CC for *Connection Counter*) MAY be notified only when the TCP TPDU’s SYN flag is set, and SHOULD NOT be present when the SYN flag is reset. Once notified, a CC or CCECHO option SHOULD be included by both TCP peers in all TPDUs on the connection. If a TCP peer has not received a CC/CCNEW option from its remote peer in a TPDU with the SYN flag set, it SHALL NOT send non-SYN TPDUs containing any of the CC options.

Effective use of the transaction support capability requires bilateral agreement between the TCP peers.

4.2.6.9 Alternate checksum (experimental, obsolete)

RFC1146 specifies an experimental mechanism for specifying and conveying alternate TPDU checksums as replacements for the standard TCP checksum conveyed in the TCP TPDU header. After bilateral negotiation via TPDUs with the SYN flag set, containing the ACR request/reply option, and if the two conveyed ACR options specified identical checksum algorithms, then each TCP peer uses that algorithm for the remainder of the TCP connection. If that algorithm provides a checksum of more than 2B, then each TPDU not containing an ACR option SHALL contain an ACD option to convey that portion of the checksum that is not conveyed in the TCP TPDU’s header’s checksum field.

Negotiation of use of a checksum algorithm other than the default TCP checksum algorithm is bilateral, requiring the peer TCP entity's consent via a returned ACD option specifying the same alternate checksum algorithm.

4.2.6.10 MD5 signature (obsolete)

RFC2385 specifies an optional mechanism for protecting TCP TPDUs with a somewhat weak message digest provided as a digital signature. When required by a TCP peer, TPDUs not containing an MD5 signature, or containing one that does not validate when using the presumably shared message digest postfix string, are ignored. Although the preceding discussion indicates that selection of this option is bilateral, its selection is by means other than TCP option negotiation and so is not addressed here.

NOTE The initial focus of the MD5 signature mechanism was protection of TCP connections for BGP sessions. Presumably, a host requiring this option for message acceptance will not be encountered during EDSA robustness testing.

4.2.6.11 User timeout

RFC5482 specifies an optional user timeout different than the default TCP timeout of the device. The option involves both advertisement of the capability and enabling the action. See the RFC for details.

4.2.6.12 Authentication option

RFC5925 specifies the use of cryptographic message authentication codes (MACs) as message digests in support of TPDU integrity and to defend against replay attacks. See the RFC for details.

4.3 Mandatory and optional protocol features

The mandatory features of the TCP protocol are

M1) The TPDU SHALL include the entire TCP header (e.g., a minimum of 20 octets, plus any option-field octets as specified by the value of the TPDU’s Data Offset field).

M2) The TPDU’s Data Offset field SHALL have a value between 5 and 15, inclusive.

M3) The TPDU’s Destination port field SHALL specify a port associated with the intended application protocol, and not a reserved port per http://www.iana.org/assignments/port-numbers.

M4) A computed checksum that has the value zero SHALL be represented in the TCP header as -0 (minus zero, 0xFFFF).

M5) TPDUs received with a checksum value that differs from a checksum computed equivalently across the virtual TCP TPDU of Figure 5 or Figure 6, as appropriate, (after virtually zeroing the TPDU’s checksum field), SHALL be discarded.

NOTE 1 TPDUs received with a checksum value of +0 (plus zero, 0x0000) always are erroneous.

M6) TCP SHALL pass IP-layer options semi-transparently between the network and application layers, per RFC1122, 4.2.3.8. IP-layer source routing options SHALL be used to determine return routes, and updated source routes SHALL update the return routes, per RFC1122, 4.2.3.8.

M7) TCP SHALL pass to its associated application user all ICMP error messages it receives from the network layer, per RFC1122, 4.2.3.9.

M8) A received TCP TPDU whose source address is an IP multicast or broadcast address SHALL be discarded without notification, per RFC1122, 4.2.3.10.

M9) A received TCP TPDU whose source address is an IP address of the receiving device SHOULD be discarded except during deliberate loopback testing.

M10) A TCP server SHALL support congestion control as specified by RFC2988 and RFC5681.

M11) The MSSN option of 4.2.6.2 shall be supported on receipt.

M12) A received MSSN, WSN, SACKN, POCR or ACR option SHALL be ignored when the TPDU’s SYN flag is reset.

M13) A received SACK, POCSP, ACD, MD5 or CC* option SHALL be ignored when the TPDU’s SYN flag is reset unless use of the option was previously negotiated or permitted by TPDUs of the same TCP connection whose SYN flag was set.

M14) A TCP server that supports one or more of the ECHORQ/ECHORP, TS, WSN, SACKN/SACK, POCR/POCSP and CC* groups of related options SHALL conform to the requirements of the relevant RFCs. For example, when the TS option is employed, the sender SHALL originate timestamp values based on a monotonic clock that progresses approximately linearly with time, in accordance with RFC1323. Also for example, when a device supports the WSN option, it SHALL include the WSN option in TCP TPDUs in which the SYN flag is set, whether or not the sending device itself has a need to employ that option.

The optional features of the TCP protocol are

C1) The initial sequence number of a TCP connection, provided during the initial connection “handshake”, SHOULD be a 32-bit value that is unpredictable to an attacker.

NOTE 2 RFC793, 3.3 required that initial sequence numbers be based on a pseudo-clock. RFC1122, 4.2.2.9 repeated this requirement. This proved to be an assist to attackers, enabling spoofed connections, so modern implementations revise this requirement to one of sequence number unpredictability.

C2) A TCP server SHOULD defer allocating state until it receives the 3rd phase TPDU of a TCP connection request, encoding a keyed hash of the requesting remote IP address and remote port as the server's initial sequence number returned in the 2nd TPDU, so that the 3rd phase TPDU can be validated if it is received, after which the server's TCP state for the connection can be allocated.

NOTE 3 Use of the CC or CCNEW option accelerates the timing of this 3rd phase during TCP connection establishment.

C3) A TCP server MAY support one or more of the ECHORQ/ECHORP, TS, WSN, SACKN/SACK, POCR/POCSP and CC* groups of related TCP options.

C4) A TCP server that supports the related SACKN/SACK options SHOULD conform to RFC2883 and, where non-contradictory, RFC2018.

C5) A TCP server MAY comply with RFC3390 and/or RFC2385.

5 Elements of other protocols required for the testing

5.1 Protocol(s) from inferior layers used by this protocol

The TCP protocol of RFC768 is specified to operate over IPv4, because the virtual TCP TPDUs of Figure 5 uses information from the conveying IPv4 NPDU. The destination-unreachable, time-exceeded and parameter-problem ICMP error PDUs of RFC792 (as amended) are used to report error conditions in received TCP TPDUs that are detected by the DUT's TCP implementation.

NOTE For TCP robustness testing, there is no requirement for the DUT to be able to receive ICMP PDUs, or to transmit ICMP PDUs of types other than the three just enumerated.

RFC2460, 8.1 specifies how TCP is adapted to work over IPv6, by using the replacement virtual TCP TPDU of Figure 6 that uses information from the conveying IPv6 NPDU. For IPv6, RFC4443 (as amended) is the corresponding controlling ICMP specification.

5.2 Protocol(s) from superior layers used to test this protocol

5.2.1 General

In principle, any DUT-implemented TCP-based server protocol that provides an APDU pseudo-echo facility can be used to explore the ability of the DUT's TCP implementation to withstand attack. In this specification, the TCP Echo server protocol on port 7 (RFC862) is used to cause the DUT to have outstanding unacknowledged transmissions, which provides a basis for testing the robustness to attack of that part of the DUT's TCP sequence number logic that would be otherwise shielded from attack.

If the TCP Echo server protocol is not available within the DUT, then robustness testing of the DUT's TCP implementation is limited to attempts to hang the DUT without interacting with the DUT, which exposes only a fraction of the DUT's potential vulnerabilities to attack. In such a case, where the TCP Echo server protocol is not available within the DUT, the test results SHALL state that fact, and that as a consequence robustness of the DUT's TCP implementation to sequence-number-based attacks was only partially testable.

6 Robustness testing

6.1 Goals that drive testing requirements

The goal of the tests described in this document is to assess:

a) the robustness of an embedded control device with an implemented set of protocols, and

b) the device's resistance to attack, including the impact on the device's reporting and control functions while sustaining such an attack.

It is not a goal to determine the correctness of the implementation of those protocols, which would be a measure of their conformance to the requirements of the various protocol specifications.

This atypical testing goal interacts with vendor decisions to provide only partial implementations of protocols that are used within a proprietary or constrained context, such that those implementations are completely functional within the usage limits imposed by that context but are not conformant to the mandatory requirements of the controlling protocol standard.

As described by specific requirements in [EDSA-310], the consequent requirement is for this testing to

1) ascertain whether the DUT and other parts of the test configuration meet normal operational expectations before testing commences;

2) determine whether the DUT can survive receipt of invalid frames while continuing to function as expected in an automation environment; and

3) determine whether the DUT can sustain intervals of high and excessive communications load.

6.2 Testing overview

The DUT must be preconditioned to support testing by

1) meeting the requirements of [EDSA-310] for demonstrating continued correct operation during testing;

2) unblocking the TCP Echo server protocol (port 7) if that is blocked in the DUT's default operating configuration.

Robustness testing occurs in three conceptual phases that may overlap, plus a test environment preconditioning phase.

The first conceptual phase, Baseline operation, attempts to demonstrate that the selected DUT protocol suite used for testing appears to operate properly for simple test cases under low network communications load, before any protocol fuzzing or stress testing is attempted.

NOTE 1 This initial demonstration of apparently correct behavior establishes the presumption that failure during additional testing is due to vulnerabilities of the specific protocol under test, rather than other protocols in the test suite.

c) The second conceptual phase, Basic robustness testing, probes the implementation for its ability to not evidence harm due to receipt of arbitrary erroneous frames, either singly or in combination. For stateful protocols, this phase also tests the response of the implementation to various state-dependent proper and improper sequences of PDUs.

NOTE 2 This conceptual phase focuses on simple protocol robustness/fuzzing tests.

d) The third conceptual phase, Load stress testing, probes the implementation's response to high traffic rates incorporating valid PDUs. For stateful protocols, this phase also tests the response of the implementation to various state-dependent proper and improper sequences of PDUs.

NOTE 3 This conceptual phase focuses on load/performance tests, first under high but supposedly sustainable receiver network communications load, then under massive overload.

Although the robustness testing of this specification is conceptualized as occurring in distinct logical phases that progress from simple single-factor testing to more complex load testing incorporating PDUs with varying characteristics, there is no requirement that an actual robustness test process work in this ordered, sequential manner; any order of testing is permitted provided that the selected order does not lead to incorrect conclusions about robustness.

Requirement TCP.R1 – Criteria for robustness test failure

Pass or fail of basic robustness and load stress testing SHALL be determined by:

- whether or not essential functions are adequately maintained under network traffic conditions created under these tests, as defined in [CRT.Essential_functions];
- any particular conditions resulting in pass/fail mandated by the testing specified in this document.

The TCP protocol that is the subject of this specification is a stateful protocol with a simple triggered-reply mechanism. Some robustness testing of a TCP implementation, particularly of the DUT's response to erroneous acknowledgments of DUT transmissions, requires use of a higher-layer protocol with its own transaction or query/response mechanisms which use TCP as a lower-level transport protocol.

Since some TCP protocol behavior can be observed only when the DUT responds to received TCP TPDUs and communicates application data, that higher-layer protocol must be one for which the DUT acts as a server, responding to remote-originated TCP TPDU requests. Thus primary testing of the DUT's TCP implementation is achieved through use of specialized TCP Echo protocol APDUs, which are conveyed by TCP TPDUs to the DUT, and the DUT's responses.

6.3 Protocol stack used for testing

6.3.1 Protocol(s) from inferior layers used by this protocol

IP is used to convey TCP TPDUs. The virtual TCP TPDU of Figure 5 or Figure 6 uses information from the conveying IP NPDU. Although this specification presumes that IPv4 NPDUs are being conveyed by "Ethernet" DPDUs, other means of conveying TCP TPDUs are not inherently precluded.

The initial EDSA CRT tested protocols include IPv4 rather than IPv6 or other networking protocols. The version of TCP specified in RFC793 (as amended) and tested under this test specification is TCP over IPv4 – that selection impacts the address field values used in the calculation of the TCP checksum. This document also specifies the necessary adaptation of test implementations of TCP that operate over IPv6.

6.3.2 Protocol(s) from superior layers used to test this protocol

TCP is a stateful transport protocol that can be used to convey intermixed APDUs of two distinct priorities, providing application protocol identification and end-device end-point addressing, as well as detection of most non-deliberate changes to the conveyed PDU.

NOTE TCP, as specified by RFC793 and its amendments, does not provide any security against deliberate forgery or changes of conveyed APDUs.

Some aspects of TCP can be observed only through use of a higher-layer protocol with a query/reply mechanism, where higher-layer PDUs conveyed via TCP and addressed to the DUT trigger higher-layer reply PDUs conveyed via TCP and addressed to the TD.

The simplest means of testing the robustness of the DUT's TCP implementation is for the DUT to provide and the TD to use the standard TCP Echo server on port 7 (RFC862). This server accepts TCP TPDUs with well-formed headers, interchanges source and destination IP addresses and source and destination ports, additively transforms received sequence numbers into sent sequence numbers, computes the checksum of the resulting TCP TPDU, and queues the TCP TPDU to IP for transmission. The ability to send an arbitrary TCP payload to a DUT TCP Echo server and receive the identical TCP payload in return provides an adequate basis for robustness testing the sequence number logic of the DUT's TCP implementation.

Requirement TCP.R2 – Conditional test report notice of limited TCP robustness testability

If the TCP Echo server protocol is not available within the DUT, then only limited TCP robustness testing is possible. In such a case the test results SHALL state the non-availability of a TCP Echo server within the DUT, and that as a consequence robustness of the DUT's TCP implementation to sequence-number-based attacks was only partially testable.

6.4 Phase 0: DUT preconditioning

Requirement TCP.R3 – Preconditioning of DUT, TD and any firewalls between the DUT and TD

a) If possible, the DUT SHALL be preconditioned for robustness testing by enabling the DUT to act as a server on the TCP Echo protocol (RFC862), possibly only for protocol test purposes, enabling that protocol within the DUT:

b) The DUT protocol stack implementation, any firewall in the DUT, and any firewalls intermediary between the DUT and the TD SHOULD be preconditioned for robustness testing by disabling any rules that block the transmission or propagation of ICMP error PDUs from the DUT to the TD;

c) The DUT, the TD(s) and possibly other devices in the test system SHALL be configured to allow observation of the performance of *essential functions* of the embedded device under the test conditions, per the requirements in [CRT.Essential_functions].

Essential functions as defined in [CRT.Essential_functions] include control loops, commands to control device configuration such as setpoints, and process alarms. A key approach to obtain observability is to use, as part of the test configuration, other automation system elements that have been engineered to communicate with and monitor the DUT.

6.5 Phase 1: Baseline operation

6.5.1 General

Requirement TCP.R4 – Demonstration of baseline operation

Before the TD commences robustness testing, the DUT shall demonstrate its ability to operate as expected in the test environment, including that the TCP component of the DUT's protocol stack is present and functioning, and that the DUT can maintain essential functions.

6.5.1.1 Initial ACK sequence number generation

The baseline operation test phase of TCP includes an assessment of the DUT's resistance to attacks based on prediction of the initial sequence numbers of "new" (e.g., spoofed) TCP connections.

Requirement TCP.R5 – Unpredictability of DUT's initial sequence numbers

Although RFC793, the initial specification of TCP, specified that initial TCP sequence numbers should be generated from a pseudo-clock, giving them a maximal-length period but making them predictable to an attacker, subsequent analysis of cyber defense needs led to a relaxation of that requirement, via RFC4987. Modern implementations of TCP SHOULD NOT generate initial sequence numbers whose values appear to be correlated.

Many tests exist that attempt to evaluate the statistical "randomness" of generates sequences of purportedly uniformly-random numbers. To provide uniformity of the assessment of randomness by different robustness test suites, the three NIST-originated STS tests – Monobit, Runs and Serial – from the GPL'd *dieharder* suite of randomness tests (see Bibliography) SHOULD be applied, with their results logged in the TCP robustness test report.

6.5.2 Presence of proprietary protocol extensions

It is common practice for vendors to extend a standard protocol in a proprietary manner to provide functionality not covered by the standard protocol, or to provide more efficient or more constrained data transport for specific device information (e.g., when multiple device parameters require atomic update or readout as a group to maintain their inter-parameter consistency). Such extensions may take the form of extra message types, extra fields in standard messages, or extra functionality for standard fields in standard messages.

NOTE Robustness testing is not required to include specialized testing of proprietary protocol extensions. Rather, vendor disclosure of such extensions is intended to provide a basis for explanation of otherwise anomalous test results.

Requirement TCP.R6 – Equipment vendor disclosure of proprietary protocol extensions

When a protocol offered for testing has been implemented with deliberate proprietary extensions, the vendor SHALL document the extensions in a manner similar to that of Clause 4, such that robustness testing can explore the intended and unintended consequences of those protocol extensions. It is acceptable that access to this proprietary information be covered by a non-disclosure agreement (NDA) between the equipment vendor and the organization that is providing the ISCI robustness testing service.

6.6 Phase 2: Basic robustness testing

6.6.1 General

Areas of specific robustness testing are identified by analysis of the controlling protocol standards. These include identification of all field value ranges and of the bounding values of the underlying message representation (e.g., a range of 10..100 in a one-byte field, whose underlying representational bounding values are 0..255). Basic robustness testing includes testing the acceptability of each of these bounding values, and of the acceptance or rejection of adjacent values to those bounding values when such adjacent values can be represented in the message encoding. It also includes testing whether fields specified to convey signed or unsigned values are distinguished and processed appropriately.

Also included in this testing are out-of-order receipt of protocol messages, and receipt of related protocol messages (e.g., multiple IP fragments of the same aggregate payload) with inconsistent options or overlapping segments and segment gaps that must be detected and handled during fragment reassembly.

Also included in this testing are various alternative representations for data elements (e.g., UTF 8 and various character escape and recoding sequences, such as the common substitution of %20 for a space character in a URL). Robustness testing of such alternatives can examine only a miniscule subset of all such possible encodings, so only a minimal random probing actually occurs.

Conceptually, basic robustness testing consists of the following, where volume or rate of message traffic is not a factor:

a) tests of valid message traffic:
   1) in expected sequences, sent at a low rate;

      NOTE TCP traffic is stateful, as may be conveyed higher-layer protocol traffic used for the testing.

   2) in unexpected but valid sequences sent at a low rate (i.e., where the messages would be considered valid for the protocol under some conditions, but are not expected for the particular protocol state, message sequence or relative time);
b) tests of low rate erroneous message traffic (e.g., the ability of the device to function after receiving erroneous messages), including:
   1) single erroneous messages, including messages with inconsistent field values;
   2) properly formed messages in erroneous sequences
   3) sequences of erroneous messages

[EDSA-310] describes the criteria for adequate performance of device essential functions under these network traffic conditions. These criteria depend upon the specific function as well as whether the function operates on the same network interface used for test traffic.

6.6.2 Basis for TCP robustness testing

Correctly and incorrectly formed TCP TPDUs sent to the DUT from the TD, some of which solicit expected responses from the DUT, form the basis for TCP robustness testing.

Requirement TCP.R7 – Testing of each message field for sensitivity to invalid content

For basic robustness testing requiring erroneous messages or message sequences, valid TCP TPDUs or TPDU sequences from the TD to the DUT SHALL be altered so that one component of the TCP TPDU is erroneous; or so that the TCP TPDU is in violation of 4.2.4, 4.2.5, 4.2.6 or 4.3; or that it is both erroneous and in violation.

Such alterations SHALL be applied to each field of the TCP TPDU where alteration might have an impact on the DUT.

During basic robustness testing, lower level PDUs employed to convey a protocol under test SHALL be valid.

NOTE 1 This type of testing can be described as single-message protocol "fuzzing".

NOTE 2 It is the TCP protocol itself that is being tested, not any conveyed higher-level application protocol whose reply mechanism(s) provide a means of forcing the DUT to send reply TPDUs containing TCP-user data to the TD, which in turn have the desired side effect of advancing the TPDU sequence number for data from the DUT to the TD.

It is suggested that basic robustness testing proceed in stages, from simple to complex, as enumerated in 6.6.1 and indicated by the following list. In general, such ordering simplifies the task of locating the source(s) of software or hardware problems should they be uncovered by the testing. However, such ordering is not a requirement.

Requirement TCP.R8 – Constituent elements in basic robustness tests

Basic TCP robustness testing SHALL include the following elements, either in distinct test phases or intermixed in a form of the test supplier's choosing:

a) Valid message traffic

b) Erroneous messages

6.7 Phase 3: Load stress testing

6.7.1 General

NOTE 1 This testing phase is used to ascertain resistance to busy plant conditions as well as deliberate attacks.

Conceptually, load stress testing consists of tests of valid message traffic sent in two distinct phases:

Phase 1 – Valid message traffic is sent at a high rate less than the saturation rate threshold specified by the DUT vendor (e.g., simulating normal but busy plant conditions);

Phase 2 – Valid message traffic is sent at up to the full auto-negotiated link rate (e.g., simulating an attack or malfunction of some kind);

Attacks against a protocol implementation take the form of repeated probing by malformed messages, or by correctly formed messages whose arrival sequence and relative timing are controlled by the attacker, or (more usually) by combinations thereof, all with the intent of exploiting some oversight or error in the specific protocol implementation(s), or of activating some intertwining aspects of a multi-layer protocol stack that were unconsidered by the implementing organization.

NOTE 2 Self-induced accidental attacks are also possible, due to designer or operator oversight.

Common examples of exploited oversights and errors are deliberate buffer overflows where the implementer had neglected to detect excessive message or field size, or recursive activation of character escape encoding when the implementer had not considered recursion. Implementation interactions within a multi-layer protocol stack may occur when an initial resource allocation (e.g., memory buffering) made by one protocol layer implementation is driven into an adjustment phase that conflicts with a resource allocation already made by a paired protocol layer implementation.

6.7.2 Basis for load stress testing

Device defenses against high traffic rates impact load stress testing, and are documented by the device vendor per the following requirement.

Requirement TCP.R9 – Documentation of self-protective rate limiting behavior

Where the DUT vendor imposes rate limiting on one or more of the protocols in the test process (e.g., "Ethernet", IP or TCP), the DUT vendor SHALL document that rate limiting occurs for that identified protocol when message rates exceed a perhaps-unspecified rate, as required by [CRT.Rate_limiting].

NOTE 1 The "Ethernet" protocol is included in this list as an identifiable placeholder for any physical and data-link protocols used to convey IPv4 or IPv6 NPDUs.

Requirement TCP.R10 – Constituent elements in load stress tests

Load stress testing SHALL include the following elements, either in distinct test phases or intermixed in a form of the test supplier's choosing:

a) high-rate valid message traffic

b) over-saturation-rate version of a), at the maximum auto-negotiated link rate that the TD can support.

Requirement TCP.R11 – Testing of saturation rate-limiting mechanism(s)

Saturation rate testing SHOULD be for a duration of two minutes, long enough for any saturation effects to manifest. Tests that inherently involve a large number of TPDUs, such as port scans, may need to run for much longer durations so that they do not cause other untoward impact on the test environment, which inherently involves the DUT, the TD and any other devices used in ascertaining the continuing performance of the DUT's other normal functionality (e.g., interactions with superior or peer automation system components).

Requirement TCP.R12 – Reproducibility of robustness testing

Basic robustness testing SHALL use a deterministic selection process (which SHALL be a seeded pseudo-random process where applicable), that tests combinations of valid and erroneous messages. See Clause 7 for specific required test cases.

Load stress testing SHALL use a deterministic selection process (which SHALL be a seeded pseudo-random process where applicable), that tests series of valid messages, with the latter including malformed messages. See Clause 7 for specific required test cases.

NOTE 2 The above constraint to use a deterministic selection process does not prohibit use of feedback from analysis of DUT responses (and non-responses) as a means of further varying and focusing testing. Nor does it prohibit use of tester-selectable options and modes to determine the aggressiveness of the test process. Rather, it is merely an attempt to facilitate reproducibility by requiring use of reproducible means to select the order, sequence and components of each test.

6.7.3 Specific load stress testing

6.7.3.1 General

Due to its complex somewhat-stateful nature and the multi-decade history of successful cyber attacks against TCP, there are a number of specific TCP protocol features, and combinations thereof, that require special attention:

a) Initial ACK sequence number generation

b) Priority traffic interleaving and de-interleaving

c) Correct handling of extermal port numbers 0 and 65535

d) DUT response to closing and opening its peer's TCP receive window

e) DUT response to inconsistent advance of its peer's TCP receive window edges

f) Peer dishonoring of the DUT's TCP receive window edges

g) Rejection of spoofed source addresses that attempt to cause the DUT to connect to itself

h) Mechanisms to withstand a SYN flood attack

i) Mechanisms to withstand a SYN/ACK flood attack

j) Mechanisms to withstand RST and similar flood attacks

k) Mechanisms to withstand spoofing of FIN, RST and similar TCP flags of a DUT correspondent

l) DUT response to manipulation of the echoed timestamp value in a TS option to disrupt the DUT's round-trip transit delay estimation process, when the DUT employs the TS option

m) DUT response to inconsistent and invalid selective acknowledgements when the DUT has negotiated the use of the SACK option

Many TCP options have constraints on their utility, which are summarized in https://www.ietf.org/mail-archive/web/tcpm/current/msg03199.html. Robustness testing should include sending such options to the DUT when they are not useful and thus likely not expected. It should also include sending options that violate any specified constraints on the option value or coding (e.g., using a timeout value of zero in the UTO option).

6.7.3.2 Priority traffic interleaving and de-interleaving

Requirement TCP.R13 – Ability to accept priority traffic

A TCP connection can convey both normal and urgent byte streams, which are interleaved before transmission and de-interleaved (i.e., separated) after reception. Thus TCP robustness testing SHALL include testing of whether the DUT fails upon receipt of TPDUs containing

- one octet of priority data, together with octets of non-priority data;
- only priority data.

NOTE Testing of other combinations (or ratios) of priority and non-priority data is permitted and encouraged.

6.7.3.3 Handling of extremal port numbers 0 and 65535

Requirement TCP.R14 – Handling of extermal port numbers 0 and 65535

TCP port numbers are unsigned 16-bit numbers with no range restrictions. Implementations that do not declare the port number as an unsigned number are expected to malfunction when a port of 32768 or greater is specified. Similarly, an implementation that neglects to consider the usability of port zero, which is seldom used, also may malfunction in a detectable manner.

6.7.3.4 DUT response to closing and opening its peer's TCP receive window

Each side of a TCP connection specifies to its TCP peer, through the sender's specified receive window, the amount of additional TCP payload data which that side is prepared to accept. Each side has complete control over its own receive window and can close, open, and reclose that window at will.

When the TD closes its receive window, the DUT SHALL honor that closure, after it is detected, and not send additional octets of TCP payload until the TD's receive window is again opened. However, due to concurrency and transmission delays, it may take some time before the remote TCP peer detects the receive window closure; during that time addition octets may be received by the TD even though it considers its receive window closed.

When the TD opens a closed receive window, the DUT SHALL honor that opening, after it is detected, and send additional octets of TCP payload up to but not greater than the number of octets specified for the open receive window. When successive TCP TPDUs from the TD alter the value of that receive window, the DUT SHALL honor them after they are detected; this can be verified to some extent by noting fluctuations in the sequence numbers of the octets that the DUT transmits in response to those changing edges of the TD's receive window.

6.7.3.5 DUT response to inconsistent advance of its peer's TCP receive window edges

Requirement TCP.R15 – Defense against inconsistent peer receive window edges

An attacker that is inconsistent in its advance of its TCP receive window edges may be able to exploit a fault in the DUT's implementation of TCP, particularly when combined with resetting of the TCP connection (via inclusion of a set RST flag in the TCP TPDU). The TD SHOULD probe the DUT's response to combinations of receive sequence number and window edge changes and use of the RST flag.

6.7.3.6 Peer dishonoring of the DUT's TCP receive window edges

Requirement TCP.R16 – Defense against peer dishonoring of the DUT's TCP receive window edges

An attacker that dishonors the remote receive window edges as declared by the DUT in its TCP TPDUs, by responding with TCP TPDUs containing payload octets with impermissible sequence numbers, often when combined with use of the URG or RST flag, MAY be able to exploit a fault in the DUT's implementation of TCP. The TD SHOULD probe the DUT's response to sent TCP TPDUs that exploit such combinations.

6.7.3.7 Rejection of spoofed source addresses that attempt to cause the DUT to connect to itself

Requirement TCP.R17 – Rejection of TCP connection attempts where the source IP address is an IP address of the DUT

Through spoofing of a source IP address, an attacker can request that a DUT establish a TCP connection with itself, either on a single DUT TCP port (i.e., destination port and spoofed source port are identical) or between a pair of different DUT TCP ports.

NOTE Historically, many TCP implementations have crashed when such a looped-to-self connection is requested. This type of attack is known as a "Land" or "LaTierra" attack.

Where the DUT able to establish such a connection, for embedded devices it usually necessarily would be a connection between two DUT ports both acting as servers. In such a situation, if the connection were to be created, the DUT might then maintain the connection (with itself) indefinitely even if there is minimal (or no) activity on the connection. The simplest way for the DUT to protect itself from this attack is to refuse to establish or maintain any connection in which the remote correspondent has an IP address assigned to the DUT. This protection could be provided by the DUT's TCP implementation, or more broadly by the DUT's IP implementation.

The TD SHALL attempt to establish a TCP connection with the DUT through use of the DUT's own IP address as the spoofed source IP address of the TD's connection attempt. Any spoofed source port may be specified, including the same port as the destination port of the TCP connection request. The TD shall report whether such a connection request appears to have succeeded, as inferred through eavesdropped observation of the DUT's IP transmissions.

6.7.3.8 Mechanisms to withstand a SYN flood attack

Requirement TCP.R18 – Resistance to SYN flood attacks

RFC4987 and subsequent analyses have determined that a TCP server implementation can defer allocating state for requested TCP connections until after the 3rd phase of the 3-way handshake that initiates a TCP connection, which corresponds to receipt of the second TPDU (an ACK) from the remote client that is attempting to initiate the connection. The basic method employed to make this deferral is to use as the initial server sequence number a keyed hash of the requesting client's source IP address and port, such that the resulting sequence number is unpredictable to an attacker but can be used by the server to validate receipt of any received TPDU that appears to be the 3rd phase of the TCP connection establishment 3-way handshake, at which point the source IP and port addresses of that 3rd TPDU can be used to provide the necessary state information to initialize a server connection record.

All TCP server implementations SHOULD implement such delayed state allocation to minimize the impact of TCP SYN flood attacks, where an attacker simply sends myriad TCP connection establishment requests without completing the 3-way handshake for any of them.

6.7.3.9 Mechanisms to withstand a SYN/ACK flood attack

Requirement TCP.R19 – Resistance to attacks that leave connections open but unused

Recent analyses have determined that a TCP attacker can employ a similar technique to defer allocating state for requested TCP connections until after connection establishment is completed. The basic method employed to make this deferral is to use as the initial client (e.g., attacker) sequence number a 4-octet value from which the attacker can generate the six octets of its spoofed source IP address and port, such that the resulting sequence number can be used by the attacker to validate receipt of any received TPDU that appears to be the 2nd phase of the TCP connection establishment 3-way handshake, at which point the source IP and port addresses of that 2nd TPDU can be used to provide the necessary state information to create and send the 3rd TPDU of the 3-way handshake, thereby saddling the server with connection state while the attacker needs none unless and until the attacker actually sends TCP payload data on the connection.

NOTE 1 This attack technique was developed to enable attackers to delay their own resource allocation until they had elicited a response from a listening TCP server, thus facilitating “shotgun” attacks on a large number of potentially responsive destination IP addresses.

NOTE 2 The above-described attacker technique can be extended by using fewer than 32 bits of the sequence number to determine the 48 bits of spoofed address, with the remaining low-order bits of the sequence number allowed to increment for each octet of TCP payload data actually transmitted by the attacker to the client. This permits the attacker to keep the TCP connection alive for a while even when the attacked server mounts defensive measures that result in the termination of inactive TCP connections, all without requiring the attacker to dedicate any memory resources for TCP state to the attacking connection.

A robust TCP server implementation that is running low on memory resources for TCP state records due to too many successful TCP connection attempts MAY terminate inactive TCP connections as a response to mitigate this potential attack. In such a case, the selection of which connection should be terminated SHOULD consider both the duration of inactivity and the recent or total amount of information transferred on the connection, preferring a connection that has been less active and/or that has been inactive longer. Under such overload conditions, other defensive measures, such as limiting the number of open TCP connections which share a specific source IP address range, also MAY be employed, provided that the DUT vendor documents both the rationale for such measures and the specific measures used to initiate the defensive action.

6.7.3.10 Mechanisms to withstand RST and similar flood attacks

Requirement TCP.R20 – Resistance to attacks that spoof TCP flags

An attacker that can eavesdrop on an existing unsecured TCP connection with the DUT can spoof the DUT’s correspondent and cause the DUT to accept the attacker’s TCP TPDUs as if they had come from the DUT’s correspondent. This creates a situation analogous to that which occurs when the attacker has itself established a TCP connection with the DUT, except that it has the added benefit of confusing the DUT’s correspondent in addition to the DUT.

An attacker that establishes and maintains a TCP connection with the DUT can induce whatever load on the DUT that the specific application protocol conveyed by the connection permits, including causing the DUT to respond to received TCP TPDUs that reset the connection (i.e., TPDUs with the RST flag set), or that open and close the attacker’s receive window. In general, there is no way within TCP to defend against such attacks, as they mimic expected TCP usage by the DUT’s correspondents. However, as with 6.7.3.9, the DUT MAY activate protective measures if it finds itself to be resource-limited, including limiting the rate at which it responds to TCP connections that have remote endpoints not in some configured table of expected correspondents, or that employ application protocols that the DUT does not consider to be essential to its deployed function.

6.7.3.11 Mechanisms to withstand spoofing of FIN, RST and similar TCP flags of a DUT correspondent

An attacker that can eavesdrop on an existing unsecured TCP connection with the DUT can spoof the DUT's correspondent and cause the DUT to accept the attacker's TCP TPDUs as if they had come from the DUT's correspondent. This creates a situation analogous to that which occurs when the attacker has itself established a TCP connection with the DUT, except that it has the added benefit of confusing the DUT's correspondent in addition to the DUT.

An attacker that spoofs the DUT's correspondent in an existing TCP connection can cause the DUT to make state transitions that are inconsistent with those expected by the DUT's correspondent. In particular, a spoofed TCP TPDU apparently from the correspondent that contains a set RST or FIN flag will cause the DUT to assume a state inconsistent with that of its correspondent.

The TD SHALL test the DUT's response to such spoofed TCP TPDUs by acting as both the DUT correspondent and a spoofing attacker, observing the DUT's response to receipt of interleaved TCP TPDUs from the "correspondent" and the "attacker". The TD SHALL test various combinations of set and reset TCP flags, including situations where the FIN flag is set and the ACK and/or PSH flags are reset.

6.7.3.12 DUT response to receipt of inconsistent or contextually-inappropriate selective acknowledgment block edges

Requirement TCP.R21 – Defense against deliberately inconsistent and/or contextually-inappropriate selective acknowledgment block edges when the DUT has negotiated the use of the SACK option

When the DUT appears to support selective acknowledgment, by validly confirming an offered SACKN option enabling use of selective acknowledgment on the TCP connection, the TD SHALL test the DUT's defense against illogical and inconsistent field values within SACK options (e.g., generally involving violations of the combined rules and requirements of RFC2018 and RFC2883).

Such testing also SHALL include deliberately inconsistent interactions between selective acknowledgments, as conveyed by SACK options, and full acknowledgments, as signaled by the TPDU ACK flag.

An attacker that deliberately violates the requirements of RFC2018 and/or RFC2883 may be able to exploit a fault in the DUT's implementation of TCP, particularly when combined with resetting of the TCP connection (via inclusion of a set RST flag in the TCP TPDU). The TD SHOULD probe the DUT's response to receipt of combinations of inconsistent and/or erroneous block subsegments as conveyed in TCP selective acknowledgment options, particularly when coupled with use of the RST flag.

6.7.3.13 DUT response to manipulation of timestamp values in a TS option

Requirement TCP.R22 – Resistance to manipulation of timestamp values in a TS option

Many TCP implementations employ non-trivial algorithms to filter the jitter from the sequence of round-trip transit delay measurements that arise from use of the TS option. Also, the TS option often is used to detect wrap-around in the remote sender's TCP sequence number space, as described in the so-called PAWS (protect against wrapped sequence numbers) part of RFC1323.

The TD SHALL test the DUT's response to deliberate manipulation of

- the time values in TS option fields, and/or
- the presence or absence of the TS option field,

where that manipulation is intended to disrupt the DUT's round-trip transit delay estimation process and/or disrupt the DUT's rejection of TPDUs that do not belong to the current half-cycle of the current TCP connection's sequence number space.

NOTE The first of these intended disruptions requires manipulation of the echoed timestamp, which RFC1323, 3.2 calls the "TS Echo reply (TSecr)", whereas the second of these intended disruptions requires misstatement of the originated timestamp, which RFC1323, 3.2 calls the "TS value (TSval)".

6.8 Reproducibility

Requirement TCP.R23 – Overall reproducibility

Baseline operation, basic robustness testing, and load stress testing SHALL be reproducible per the requirements of [CRT.Reproducibility].

Those requirements recognize that deterministic behavior of the DUT itself is not under the control of the tester and must be assumed. Further, it is acceptable to branch a test process based upon prior results. Thus a change to the DUT may impact repeatability of a test even if the change does not intentionally cause variance for that test.

7 Specific test cases

Requirement TCP.R24 – Specific test cases

The tested suite of protocols SHALL be documented in at least the detail specified by Table 5.

Table 5 – Protocols used in test process

Protocol layer tested	Permissible alternatives	Protocols tested	Maximum network communications load at which deliberate limiting occurs
Physical layer	IEEE 802.3		
Data-link layer	"Ethernet"		
Network layer	IPv4 + ICMPv4 error reporting or IPv6 + ICMPv6 error reporting		
Transport layer	TCP		
Application layer	TCP Echo		

Requirement TCP.R25 – Testing SHALL include at least that specified by Table 6 through Table 30

These tables are descriptive, not proscriptive – there is no requirement that conforming robustness testing actually employ test sequences that are ordered or grouped as described in these tables. Tests where ***Results*** are indicated as Pass or Fail, SHALL pass if the indicated ***Expected response*** is observed. If the Results row in a table does not indicate "Pass/Fail," this means that the test results provide security-relevant information about the DUT to be included in the test report, but cannot cause a device to fail certification.

Table 6 – TCP.T00: Baseline operation

Test ID	TCP.T00
Test name	Baseline operation
Test description	The basic operational aspects of the protocol under test, and of any inferior or selected superior supporting protocols used in the testing, shall be demonstrated as a means of checking that gross configuration or other errors are not interfering with the testing process, that TCP is a functioning part of the DUT's protocol stack, and that the protocol implementation under test performs approximately as expected when not under test
Reference requirements	Requirement TCP.R4
Test type	Baseline operation
Test status	Mandatory
Expected DUT behavior	The DUT demonstrates basic protocol operability in the test configuration
Test object	To validate the lack of major errors in the configuration of the DUT and test environment
Test configuration	A TD is connected to the DUT by an underlying switched network that uses IEEE 802 and IP addressing, as specified in [CRT.Test_configuration_1]
Test procedure	The TD establishes that DUT is reachable and functions normally in the test environment, before protocol-specific testing commences. This includes establishing that those protocol components of the DUT that are used during TCP robustness testing are functional and accessible. For each application layer protocol that will be used for robustness testing, the TD establishes a TCP connection with the DUT over the selected network layer (usually IPv4) and establishes that the selected protocol stack is, to at least some extent, usable
Expected DUT response	The DUT demonstrates expected behavior in its "automation" environment, including that the TCP component of the protocol stack is present and functioning and that the DUT can adequately maintain essential functions
Ultimate results	Pass or fail
Remarks	Initial failure of this test indicates a probable problem with the configuration of the TD or the test environment

Table 7 – TCP.T01: Statistical analysis of initial sequence number values

Test ID	TCP.T01
Test name	Statistical analysis of initial sequence number values
Test description	The TD repeatedly initiates TCP connections and records the returned initial sequence number values generated by the DUT. At test completion, a statistical analysis is made of those sequence numbers and any apparent clustering or correlation in their values is reported
Reference requirements	Requirement TCP.R7, violating 4.3, M11; Requirement TCP.R5
Test type	Baseline operation
Test status	Mandatory
Expected DUT behavior	The DUT generates initial sequence numbers that are not predictable to an attacker
Test object	To probe the robustness of the DUT's resistance to spoofed connections
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD attempts to establish TCP connections with the DUT and records the initial sequence numbers returned by the DUT during connection establishment. The TD assesses the correlation of those initial DUT sequence numbers via use of the three tests STS Monobit, STS Runs, and STS Serial from the GPL'd *dieharder* random number test suite (see Bibliography). At a minimum, 100,000 samples or 2 hours of sample generation SHALL be used, whichever comes first.
Expected DUT response	The DUT SHOULD establish connections with sequence numbers that do not have any apparent correlation or predictability
Results	Quality measure for the sequence of generated random numbers
Remarks	This test exposes failures to generate apparently uncorrelated initial sequence numbers, which makes the DUT vulnerable to TCP connection spoofing. This test can be combined with other DUT TCP testing, including the various TCP flood tests

Table 8 – TCP.T02: Truncated TPDU: truncated fixed header

Test ID	TCP.T02
Test name	Truncated TPDU: truncated fixed header
Test description	A truncated TCP TPDU is sent as an IP NPDU payload, where the payload is the initial octets of a correctly formed TCP TPDU but where the total size of the payload is between 0 and 19 octets, inclusive, and thus less than the minimum TCP header size of 20 octets
Reference requirements	Requirement TCP.R7, violating 4.3, M1 and M3
Test type	Basic robustness: PDU structural violations
Test status	Mandatory
Expected DUT behavior	The DUT checks the TPDU's length – the reported size of the NPDU's payload – before checksum validation. The minimum required NPDU payload length is 20 octets
Test object	To probe the robustness of the DUT's parsing of TCP TPDUs and protection against malformed TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s). The TD MAY monitor for any response from the DUT
Test procedure	The TD sends the initial octets of an otherwise-valid TCP TPDU, where the conveying NPDU's payload is truncated to less than 20 octets. The TD monitors for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	The DUT is expected to reply with an ICMP ParameterProblem (type 0x04) PDU

Table 9 – TCP.T03: Truncated TPDU: truncated header options

Test ID	TCP.T03
Test name	Truncated TPDU: truncated header options
Test description	A truncated TCP TPDU is sent as an IP NPDU payload, where the payload is the initial 20 or more octets of a correctly formed TCP TPDU, but where truncation occurs within a non-null header option field whose existence is indicated by a TPDU DataOffset field whose value is greater than 5.
Reference requirements	Requirement TCP.R7, violating 4.3, M2
Test type	Basic robustness: PDU structural or content semantic violations
Test status	Mandatory
Expected DUT behavior	The DUT checks the TPDU's checksum and then the TPDU's length – the reported size of the NPDU's payload – for adequacy and consistency with the values of TPDU header fields before processing of any conveyed TPDU payload
Test object	To probe the robustness of the DUT's parsing of TCP TPDUs and protection against malformed TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s). The TD MAY monitor for any response from the DUT
Test procedure	The TD sends the initial octets of an otherwise-valid TCP TPDU, where the conveying NPDU's payload is truncated to a number of octets of 20 or greater but less than the value of the TPDU's DataOffset field times 4. The TD monitors for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	The DUT is expected to reply with an ICMP ParameterProblem (type 0x04) PDU

Table 10 – TCP.T04: Truncated TPDU: truncated priority data

Test ID	TCP.T04
Test name	Truncated TPDU: truncated priority data
Test description	A truncated TCP TPDU is sent as an IP NPDU payload, where the payload is the initial octets of a correctly formed TCP TPDU containing priority data (as indicated by a set TPDU URG flag), but where truncation occurs after the header field but before the end of the priority data field. In other words, the conveying NPDU payload length is within the range [(TPDU.DataOffset × 4) .. (TPDU.DataOffset × 4 +TPDU.UrgentPointer)]
Reference requirements	Requirement TCP.R7, violating 4.3, M2
Test type	Basic robustness: PDU content semantic violations
Test status	Mandatory
Expected DUT behavior	The DUT checks the TPDU's checksum and then the TPDU's length – the reported size of the NPDU's payload – for adequacy and consistency with the values of TPDU header fields before processing of any conveyed TPDU payload
Test object	To probe the robustness of the DUT's parsing of TCP TPDUs and protection against malformed TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends the initial octets of an otherwise-valid TCP TPDU with a set URG flag, where the conveying NPDU's payload is truncated to a number of octets in the range [(TPDU.DataOffset × 4) .. (TPDU.DataOffset × 4 +TPDU.UrgentPointer)]. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	The DUT is expected to reply with an ICMP ParameterProblem (type 0x04) PDU

Table 11 – TCP.T05: Invalid TPDU checksum

Test ID	TCP.T05
Test name	Invalid TPDU checksum
Test description	TCP TPDUs are sent whose checksum field value differs from the computed binary checksum for the TPDU (i.e., so that numeric +0 (0x0000) is distinct from -0 (0xFFFF))
Reference requirements	Requirement TCP.R7, violating 4.3, M4 and M5
Test type	Basic robustness: PDU content semantic violations
Test status	Mandatory
Expected DUT behavior	The DUT validates TCP TPDU checksums on receipt and compute TCP TPDU checksums prior to transmission. The value +0 is not used
Test object	To probe the robustness of the DUT's checksum processing for TCP TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends an otherwise valid TCP TPDU whose checksum field differs from a correctly computed checksum for the TPDU. The TD monitors for any response from the DUT. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	The DUT is expected to reply with an ICMP ParameterProblem (type 0x04) PDU

Table 12 – TCP.T06: Malformed or undefined TCP TPDU options

Test ID	TCP.T06
Test name	Malformed or undefined TCP TPDU options
Test description	The TD includes non-null TCP option fields in some of its TCP TPDUs. Some of these option fields are malformed, or reference undefined options. Each of the TCP options specified in 4.2.6 SHALL be included in some tests, sometimes singly and sometimes in combinations of two or more options.
Reference requirements	Requirement TCP.R7, violating 4.3, M10
Test type	Basic robustness: PDU content syntactic or semantic violations
Test status	Mandatory
Expected DUT behavior	The DUT validates the TCP option field
Test object	To probe the robustness of the DUT’s option processing for TCP TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends an otherwise valid TCP TPDU whose option field is non-null and which may, or may not, be contextually valid. The TD monitors for any response from the DUT. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	The DUT is expected to reply with an ICMP ParameterProblem (type 0x04) PDU when the option field is malformed or contains an undefined option

Table 13 – TCP.T07: Unrestricted interpretation of select TCP TPDU options

Test ID	TCP.T07
Test name	Unrestricted interpretation of select TCP TPDU options
Test description	The TD includes one or more of the TCP options MSSN, WSN, SACKN, POCR and ACR, in some TCP TPDUs where the SYN flag is reset. Testing SHALL include each of these options, both singly and in some combinations
Reference requirements	Requirement TCP.R7, violating 4.3, M12
Test type	Basic robustness: PDU content semantic violations
Test status	Mandatory
Expected DUT behavior	The DUT validates the TCP option field when present. Received TPDUs that contain any of MSSN, WSN, SACKN, POCR or ACR and which have the SYN flag reset are erroneous and are ignored
Test object	To probe the robustness of the DUT’s option processing for TCP TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends an otherwise valid TCP TPDU whose SYN flag is reset and whose option field is syntactically valid and contains one or more of the TCP options MSSN, WSN, SACKN, POCR and ACR. The content of the remainder of the TPDU is selected to provide evidence on subsequent transfers within the TCP connection of whether the TPDU was ignored that had the SYN flag reset, and that contained TPDU options that are invalid when the SYN flag is reset. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	The DUT is expected to be robust against receipt of options when in protocol states where such options are not permissible or relevant (or both)

Table 14 – TCP.T08: “Land” and “LaTierra” attacks

Test ID	TCP.T08
Test name	“Land” and “LaTierra” attacks
Test description	In a “Land” attack, the TD initiates TCP connection attempts with the DUT, spoofing the source IP address and port in the connection attempt so that they are identical to the destination IP address and port of the DUT. The goal of this attack is to induce the DUT to attempt to complete a TCP connection with itself, resending the TCP reply in response to each of its own TPDUs. In a “LaTierra” attack, the TD modifies the above procedure by sending TCP TPDUs to more than one port of the DUT, attempting to cross-link the DUT’s ports into a TCP connection where each port acts as responder, again attempting to hang the DUT’s TCP implementation in a loop
Reference requirements	Requirement TCP.R7, violating 4.3, M9 Requirement TCP.R17
Test type	Basic robustness: self-referent NPDU addressing
Test status	Mandatory
Expected DUT behavior	The DUT rejects any TCP connection request where the source IP address is an IP address of the DUT
Test object	To probe the robustness of the DUT’s ability to survive spoofed connection attempts with itself
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends an otherwise valid TCP connection establishment TPDU whose conveying NPDU specifies the DUT’s IP address as both the NPDU’s destination and source addresses. For the first such test, the source port of the connection request is identical to the destination port; for subsequent tests the two ports differ. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	1) The DUT is expected to discard the TPDU without notification 2) This test can expose failures to protect against spoofed source addresses that attempt to cause the DUT to establish a TCP connection with itself. The Bibliography cites a paper with more information about these attacks

Table 15 – TCP.T09: Spoofed TCP flags

Test ID	TCP.T09
Test name	Spoofed TCP flags
Test description	The TD establishes TCP connections with the DUT, then sends TCP TPDUs on the connection whose TCP flags have various inconsistent or illogical states, including occasional use of the reserved flag bits. Testing shall include a substantial number of such flag combinations
Reference requirements	Requirement TCP.R20
Test type	Basic robustness: PDU content semantic violations
Test status	Mandatory
Expected DUT behavior	The DUT recovers from such nonsensical intermixed TCP flags, either continuing or resetting or terminating the connection
Test object	To probe the robustness of the DUT's processing of TCP TPDUs that are inconsistent with current TCP state
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends otherwise valid TCP TPDUs whose flags are occasionally non-sensical or inconsistent with the state of the TCP connection, including occasional use of the reserved flag bits. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	It does not matter whether the DUT continues, resets or terminates the connections whose TCP flags assume inconsistent state; it only matters that the DUT does not itself lock up immediately or eventually through resource exhaustion

Table 16 – TCP.T10: Inconsistent peer receive window edges

Test ID	TCP.T10
Test name	Inconsistent peer receive window edges
Test description	The TD establishes a TCP connection with the DUT, then sends TCP TPDUs for the connection that report inconsistent and illogical receive window edges for the TD's side of the connection, combining this with intermittent reset of the connection
Reference requirements	Requirement TCP.R15
Test type	Basic robustness
Test status	Mandatory
Expected DUT behavior	The DUT survives attacks that manipulate the peer's receive window edges with inconsistent and illogical values
Test object	To probe the robustness of the DUT's defensive processing of TCP TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends otherwise valid TCP TPDUs whose receive window edges are illogical or inconsistent with respect to prior TPDUs of the same TCP connection, interleaving reset of the connection at a rate intended to induce loss of synchronization between the TCP connection endpoints. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	It does not matter whether the DUT continues, resets or terminates the connections whose TCP receive window edges are reported with inconsistent values; it only matters that the DUT does not itself lock up or irreversibly allocate TCP connection state resources to connections that are terminated in error

Table 17 – TCP.T11: Defense against peer dishonoring of the DUT's TCP receive window edges

Test ID	TCP.T11
Test name	Defense against peer dishonoring of the DUT's TCP receive window edges
Test description	The TD establishes a TCP connection with the DUT, then sends TCP TPDUs for the connection that do not honor the DUT's receive window edges by responding with TCP TPDUs containing payload octets with impermissible sequence numbers, combining this with occasional reset of the connection.
Reference requirements	Requirement TCP.R16
Test type	Basic robustness
Test status	Mandatory
Expected DUT behavior	The DUT survives attacks that dishonor the DUT's receive window edges by sending TCP payload octets assigned to inconsistent and illogical sequence number values
Test object	To probe the robustness of the DUT's defensive processing of TCP TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends otherwise valid TCP TPDUs which contain non-null payloads whose sequence numbers are inconsistent with the DUT's expected or reported receive window edges, as assessed with respect to prior TPDUs of the same TCP connection. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	It does not matter whether the DUT continues, resets or terminates the connections whose TCP receive window edges are reported with inconsistent values; it only matters that the DUT does not itself lock up or irreversibly allocate TCP connection state resources to connections that are terminated in error

Table 18 – TCP.T12: Inconsistent and/or contextually-inappropriate selective acknowledgment block edges

Test ID	TCP.T12
Test name	Inconsistent and/or contextually-inappropriate selective acknowledgment block edges
Test description	The TD establishes a TCP connection with the DUT in which use of selective acknowledgment is negotiated, then sends TCP TPDUs with the SACK option that report inconsistent and/or contextually-inappropriate selective acknowledgment block edges for the TD's side of the connection, combining this with occasional reset of the connection
Reference requirements	Requirement TCP.R21
Test type	Basic robustness
Test status	Mandatory
Expected DUT behavior	The DUT survives attacks that manipulate the peer's receive window edges with inconsistent and illogical values
Test object	To probe the robustness of the DUT's defensive processing of TCP TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	After successfully negotiating use of selective acknowledgment on a TCP connection, the TD sends otherwise valid TCP TPDUs containing SACK options, ore or more of whose block edges are inconsistent or contextually inappropriate with respect to prior TPDUs of the same TCP connection. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	It does not matter whether the DUT continues, resets or terminates the connections on which the inconsistent or contextually-inappropriate selective acknowledgment block edges are reported; it only matters that the DUT does not itself lock up or irreversibly allocate TCP connection state resources to connections that are terminated in error

Table 19 – TCP.T13: Manipulation of DUT timestamps echoed to the DUT

Test ID	TCP.T13
Test name	Manipulation of DUT timestamps echoed to the DUT
Test description	The TD establishes TCP connections with the DUT and includes the TS option in at least some of the TPDUs it sends to the DUT. If the DUT includes any TS options in the TPDUs that it sends to the TD, the TD manipulates those DUT-provided timestamp values when it is supposed to echo them, in violation of the requirements of RFC1323. The goal of this manipulation is to probe any DUT logic related to reasonableness of returned timestamps, attempting to cause the DUT to compute round-trip-delays that trigger abnormal recovery actions
Reference requirements	Requirement TCP.R22
Test type	Basic robustness
Test status	Mandatory
Expected DUT behavior	The DUT survives attacks that manipulate the values of putatively-echoed timestamps
Test object	To probe the robustness of the DUT's defensive processing of TCP TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends otherwise valid TCP TPDUs containing TS options, some of whose timestamp values are inconsistent or in violation of RFC1323. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	It does not matter whether the DUT continues, resets or terminates the connections on which the inconsistent or contextually-inappropriate selective acknowledgment block edges are reported; it only matters that the DUT does not itself lock up or irreversibly allocate TCP connection state resources to connections that are terminated in error

Table 20 – TCP.T14: Priority traffic interleaving

Test ID	TCP.T14
Test name	Priority traffic interleaving
Test description	The TD establishes TCP connections with the DUT, sending TPDUs that contain only non-priority traffic (i.e., URG flag reset) intermixed with TPDUs that contain only priority traffic (i.e., URG flag set and all octets of TPDU payload are urgent) intermixed with TPDUs that contain both priority and non-priority traffic (i.e., URG flag set, but less than all octets of TPDU payload are urgent)
Reference requirements	Requirement TCP.R13
Test type	Basic robustness: PDU parsing and semantic violations
Test status	Mandatory; however, the DUT is permitted to discard received urgent octets
Expected DUT behavior	The DUT de-interleaves received priority traffic correctly and, to the extent testable, interleaves sent priority traffic correctly
Test object	To probe the robustness of the DUT's processing of mixed urgent and normal priority octets in TCP TPDUs
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD sends to the DUT valid TCP TPDU containing various mixtures of all normal, all urgent, or mixed urgent and normal octets. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	This test exposes problems with the DUT implementation in segregating urgent and normal traffic. It does not require the DUT to implement an urgent traffic stream; however, the DUT is not permitted to fail when such an urgent traffic stream is included (perhaps by spoofing) within an existing TCP connection

Table 21 – TCP.T15: Use of extremal TCP port numbers

Test ID	TCP.T15
Test name	Use of extremal TCP port numbers
Test description	The TD attempts to establish TCP connections with the DUT on both port zero (0x0000) and port 65535 (port 0xFFFF), to determine whether the DUT properly handles these seldom-used ports as zero-origin unsigned port numbers
Reference requirements	Requirement TCP.R14
Test type	Basic robustness: PDU parsing and semantic violations
Test status	Mandatory
Expected DUT behavior	The DUT replies correctly to TPDUs addressed to these ports without causing an exception
Test object	To probe the robustness of the DUT's processing of port numbers that an implementation may not correctly support
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. ICMP error reporting by the DUT SHOULD be enabled at any intervening firewall(s)
Test procedure	The TD attempts to establish TCP connections with the DUT on valid TCP destination ports zero (0x0000) and 65535 (0xFFFF) and observes the DUT's responses. The TD MAY monitor for any response from the DUT
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	1) The DUT is expected to respond to the TCP connection establishment request TPDU. If no service is available on the specified TCP port, the expected response consists of a TCP TPDU with the RST flag set 2) A DUT implementation that neglects to treat port numbers as unsigned numbers is likely to fail this test

Table 22 – TCP.T16: TCP conveyed-application robustness

Test ID	TCP.T16
Test name	TCP conveyed-application robustness
Test description	The TD generates TCP data and sends it to open TCP ports on the DUT. Any data size supported by TCP may be generated, from 0 B to 65,535 B of TCP payload. The TCP data uses a variety of data patterns known to cause problems for some TCP-conveyable protocols, without attempting to target any specific TCP-conveyable protocol.
Reference requirements	
Test type	Basic robustness: APDU content semantic violations
Test status	Mandatory
Expected DUT behavior	The DUT continues to function while receiving such TCP TPDUs, provided that the load thus induced is less than that claimed as supportable by the DUT vendor
Test object	To probe the robustness of the DUT's ability to receive and withstand a rate-limited burst of TPDUs addressed to its discovered open ports, similar to that induced by an attacker's follow-up to a port scan
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]
Test procedure	The TD sends valid TCP TPDUs conveying varying but focused APDU data addressed to various TCP ports of the DUT, at a rate less than that at which the DUT's manufacturer claims DUT protective measures will be invoked
Expected DUT response	The DUT is expected to continue network communication even under focused load while adequately maintaining essential functions
Results	Pass or fail
Remarks	

Table 23 – TCP.T17: Maintenance of service under high load: Raw TPDU flood of urgent data

Test ID	TCP.T17
Test name	Maintenance of service under high load: Raw TPDU flood of urgent data
Test description	A flurry of TCP TPDUs, padded to the maximum possible size, all of which are marked as Urgent data, is sent to the DUT to attempt to overwhelm the DUT’s receive processing and storage resources. All TPDUs have the TCP URG flag set in an attempt to cause the DUT scheduler to prioritize processing of the TPDUs over other tasks. The TPDU flood rate is selected to be a high load test, not a saturation test. See [CRT.Rate_limiting] for additional requirements
Reference requirements	Requirement TCP.R10
Test type	Load stress
Test status	Mandatory
Expected DUT behavior	The DUT protects itself against a flood of received TCP TPDUs
Test object	To evaluate the DUT’s ability to receive and withstand a burst of TPDUs addressed to it
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. The DUT vendor SHALL state a rate limit below which protective measures are not expected to be invoked
Test procedure	The TD sends valid TPDUs containing a maximal amount of urgent data, addressed to the DUT, at the vendor-specified rate limit. TPDUs SHOULD be padded to the maximum size possible (according to the limiting MTU), typically, resulting in a 1460 B payload. All data in these TPDUs is marked as urgent
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	

Table 24 – TCP.T18: Maintenance of service under high load: Initial SYN flood

Test ID	TCP.T18
Test name	Maintenance of service under high load: Initial SYN flood
Test description	A large number of apparently valid TCP TPDUs that begin to initiate TCP connections which subsequently fail to complete establishment (i.e., the full 3-way TCP handshake) are sent to the DUT to attempt to overwhelm the DUT's connection state storage resources. The TPDU flood rate is selected to be a high load test, not a saturation test. See [CRT.Rate_limiting] for additional requirements
Reference requirements	Requirement TCP.R18
Test type	Load stress
Test status	Mandatory
Expected DUT behavior	The DUT protects itself against a flood of received TCP TPDUs that begin connection negotiation and then leave the 3-phase connection negotiation process hanging, waiting to complete
Test object	To evaluate the DUT's ability to receive, withstand and recover from a burst of TCP connections that are left in a half-open state, due to what is known as a SYN flood attack
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. The DUT vendor SHALL state a rate limit below which protective measures are not expected to be invoked
Test procedure	The TD sends many valid TPDUs for currently unused (IP source address, TCP source port) pairs addressed to the DUT's TCP port, with the TPDU SYN flag set, but fails to respond to the consequent TCP TPDUs that have both SYN and ACK sent. The Broadcast IP address SHALL be used as one of the IP source addresses
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	During this flood the TD MAY attempt to establish and use other TCP connections to survey the DUT's behavior, with non-null traffic on those other surveying TCP connections

Table 25 – TCP.T19: Maintenance of service under high load: Unused open connection flood

Test ID	TCP.T19
Test name	Maintenance of service under high load: Unused open connection flood
Test description	A large number of apparently valid TCP TPDUs are sent to the DUT to initiate TCP connections and to subsequently complete the establishment of those connections (i.e., the full 3-way TCP handshake), after which the connections are left open but unused with no TPDU-payload traffic, all in attempt to overwhelm the DUT's connection state storage resources.
Reference requirements	Requirement TCP.R19
Test type	Load stress
Test status	Mandatory
Expected DUT behavior	The DUT protects itself against a flood of received TCP TPDUs that complete the 3-phase connection negotiation process but fail to use the connections thus established to convey non-null TPDU payloads, by closing and abandoning those idle connections when the overload occurs
Test object	To evaluate the DUT's ability to receive, withstand and recover from a burst of TCP connections that are opened successfully but then left unused
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. The DUT vendor SHALL state a rate limit below which protective measures are not expected to be invoked
Test procedure	The TD sends many valid TPDUs for currently unused (IP source address, TCP source port) pairs addressed to the DUT's TCP port, with the TPDU SYN flag set, then responds to the consequent TCP reply TPDUs from the DUT that have both SYN and ACK flags set that are sent to confirm establishment of the TCP connections.
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	During this flood the TD MAY attempt to establish and use other TCP connections to survey the DUT's behavior, with non-null traffic on those other surveying TCP connections

Table 26 – TCP.T20: Maintenance of service under high load: Segment reassembly flood

Test ID	TCP.T20
Test name	Maintenance of service under high load: Segment reassembly flood
Test description	A number of valid TCP TPDUs for established TCP connections, each of which has a sequence number greater than the next expected sequence number, are sent to the DUT to attempt to overwhelm the DUT's storage resources or storage management. The TPDU flood rate is selected to be a high load test, not a saturation test. See [CRT.Rate_limiting] for additional requirements
Reference requirements	Requirement TCP.R19
Test type	Load stress
Test status	Mandatory
Expected DUT behavior	The DUT protects itself against a flood of received TCP TPDUs that skip portions of the TCP stream
Test object	To evaluate the DUT's ability to receive, withstand and recover from a burst of incorrectly advancing TCP sequence numbers, attempting to fill the DUT's segment reassembly buffers
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. The DUT vendor SHALL state a rate limit below which protective measures are not expected to be invoked
Test procedure	The TD establishes one or more valid TCP connections for currently unused (IP source address, TCP source port) pairs addressed to the DUT's TCP port, then repeatedly sends TCP TPDUs that have incorrectly advancing sequence numbers.
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	During this flood the TD MAY attempt to establish and use other TCP connections to survey the DUT's behavior, with non-null traffic on those other surveying TCP connections

Table 27 – TCP.T21: Maintenance of service under high load: Closed receive window flood

Test ID	TCP.T21
Test name	Maintenance of service under high load: Closed receive window flood
Test description	A number of established TCP connections each set their receive window size to zero while requesting data from the DUT. The TPDU flood rate is selected to be a high load test, not a saturation test. See [CRT.Rate_limiting] for additional requirements
Reference requirements	Requirement TCP.R19
Test type	Load stress
Test status	Mandatory
Expected DUT behavior	The DUT protects itself against a flood of received TCP TPDUs for established TCP connections that close their receive windows (by setting the receive window size to zero) at a time when they have requested that the DUT transfer information to them
Test object	To evaluate the DUT's ability to withstand and recover from a number of connections which set their receive window size to zero, forcing buffering upon the DUT in an attempt to overload the DUT's resource management
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. The DUT vendor SHALL state a rate limit below which protective measures are not expected to be invoked
Test procedure	The TD establishes one or more valid TCP connections for currently unused (IP source address, TCP source port) pairs addressed to the DUT's TCP port, then repeatedly sends TCP TPDUs that have their receive window closed (i.e., set to zero size).
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	During this flood the TD MAY attempt to establish and <u>use</u> other TCP connections to survey the DUT's behavior, with non-null traffic on those other surveying TCP connections

Table 28 – TCP.T22: Maintenance of service under high load: RST flood

Test ID	TCP.T22
Test name	Maintenance of service under high load: RST flood
Test description	A number of apparently valid TCP TPDUs for established TCP connections, each of which has its RST flag set, are sent to the DUT to attempt to overwhelm the DUT’s connection state processing and storage resources. The TPDU flood rate is selected to be a high load test, not a saturation test. See [CRT.Rate_limiting] for additional requirements
Reference requirements	Requirement TCP.R20
Test type	Load stress
Test status	Mandatory
Expected DUT behavior	The DUT protects itself against a flood of received TCP TPDUs that reset established TCP connections
Test object	To evaluate the DUT’s ability to receive, withstand and recover from a burst of reset commands for existing TCP connections, due to what is known as an RST flood attack
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. The DUT vendor SHALL state a rate limit below which protective measures are not expected to be invoked
Test procedure	The TD establishes one or more valid TCP connections for currently unused (IP source address, TCP source port) pairs addressed to the DUT‘s TCP port, then repeatedly sends TCP TPDUs that have their RST flag set to reset those connections.
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	During this flood the TD MAY attempt to establish and use other TCP connections to survey the DUT’s behavior, with non-null traffic on those other surveying TCP connections

Table 29 – TCP.T23: Maintenance of service under high load: FIN flood

Test ID	TCP.T23
Test name	Maintenance of service under high load: FIN flood
Test description	A number of apparently valid TCP TPDUs for established TCP connections, each of which has its FIN flag set, are sent to the DUT to attempt to disrupt the DUT's existing connections. The TPDU flood rate is selected to be a high load test, not a saturation test. See [CRT.Rate_limiting] for additional requirements
Reference requirements	Requirement TCP.R19, 6.7.3.11
Test type	Load stress
Test status	Mandatory
Expected DUT behavior	The DUT protects itself against a flood of received TCP TPDUs that attempt to close connections
Test object	To evaluate the DUT’s ability to receive, withstand and recover from a burst of close commands for existing TCP connections, due to what is known as a FIN flood attack
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. The DUT vendor SHALL state a rate limit below which protective measures are not expected to be invoked
Test procedure	The TD establishes one or more valid TCP connections for currently unused (IP source address, TCP source port) pairs addressed to the DUT‘s TCP port, then repeatedly sends TCP TPDUs that have their FIN flag set as if they were spoofed TPDUs that attempt to interfere with and close those connections.
Expected DUT response	The DUT continues to adequately maintain essential functions
Results	Pass or fail
Remarks	During this flood the TD MAY attempt to use those TCP connections whose FIN TPDUs are spoofed as part of the attack, as well as to establish and use other TCP connections, to survey the DUT’s behavior with non-null traffic on those TCP connections

Table 30 – TCP.T24:Maintenance of service under high load, including network saturation: Raw TPDU flood

Test ID	TCP.T24
Test name	Maintenance of service under high load, including network saturation: Raw TPDU flood
Test description	A flurry of TCP TPDUs is sent to the DUT to attempt to overwhelm the DUT's receive processing and storage resources. . This test proceeds in two phases: • Phase 1: as a high load test during which the DUT SHOULD respond normally to received messages • Phase 2: as a network saturation test during which the DUT MAY invoke protective behaviors such as blocking network reception but SHOULD otherwise function normally. See [CRT.Rate_limiting] for additional requirements
Reference requirements	Requirement TCP.R10
Test type	Load stress
Test status	Mandatory
Expected DUT behavior	The DUT protects itself against a flood of received TCP TPDUs • Phase 1: The DUT continues to function, adequately maintaining all essential functions, in the presence of a sudden burst of received TCP TPDUs, provided that the load thus induced is less than that claimed as supportable by the DUT vendor; • Phase 2: The DUT adequately maintains essential control, even if it must reduce or cease other essential functions during the period of network overload.
Test object	To evaluate the DUT's ability to receive and withstand a burst of TPDUs addressed to it
Test configuration	A TD is connected to the DUT by an underlying switched network that uses either IPv4 or IPv6 addressing, as specified in [CRT.Test_configuration_1]. The DUT vendor SHALL state a rate limit below which protective measures are not expected to be invoked
Test procedure	The TD sends valid TCP TPDUs that are either explicitly or implicitly addressed to the DUT • Phase 1: at a rate less than that at which the DUT's manufacturer claims DUT protective measures will be invoked; • Phase 2: at a rate up to the auto-negotiated maximum rate of the underlying network, maintains that high load rate for two minutes. TCP TPDUs sent to the DUT MAY be conveyed by IP packets using any of the types of explicit or implicit IP addressing (e.g., unicast, broadcast ,multicast, or (with IPv6) anycast), in any combination
Expected DUT response	• Phase 1: The DUT is expected to continue network communication even under high load while adequately maintaining essential functions. • Phase 2: The DUT is expected to activate protective measures at some (vendor unspecified) level of resource demand, and to recover some reasonable time interval after that demand for resources is reduced substantially below the level at which the protective measures were triggered. The DUT is expected to adequately maintain essential control throughout the test
Results	Pass or fail
Remarks	The DUT vendor is not required to be able to predict the messaging rate at which such protective measures are invoked, but SHOULD be able to put an upper bound on time after the stimulus ceases before the recovery is complete

Bibliography

IANA protocol and number registries, http://www.iana.org/protocols/
registries of various assigned code points for standard Internet protocols

NOTE For each RFC*nnn*, the controlling version can be found at http://tools.ietf.org/html/rfc*nnn*.

RFC1072, *TCP extensions for long-delay paths*

RFC1185, *TCP extensions for high-speed paths*

RFC1750, *Randomness recommendations for security*

RFC1948, *Defending against sequence number attacks*

RFC2581, *TCP congestion control*

RFC2827, *Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing*

RFC3704, *Ingress filtering for multihomed networks*

NIST statistical test suite, at http://csrc.nist.gov/groups/ST/toolkit/rng/stats_tests.html

Dieharder statistical test suite, at http://www.phy.duke.edu/~rgb/General/dieharder.php

Denial of service attacks: Teardrop and Land, at http://users.tkk.fi/lhuovine/study/hacker98/dos.html

Sockstress, at http://en.wikipedia.org/wiki/Sockstress

— — — — — —