Network Working Group   Brooks Hickman
Internet-Draft   Spirent Communications
Expiration Date: November 2002   David Newman
                                                    Network Test
                                                  Saldju Tadjudin
                                              Spirent Communications
                                                    Terry Martin
                                                    M2networx INC
                                                      May 2002

Benchmarking Methodology for Firewall Performance
<draft-ietf-bmwg-firewall-04.txt>

Status of this Memo

Table of Contents

[Page 1]

1. Introduction

This document provides methodologies for the performance
benchmarking of firewalls. It provides methodologies in four areas:
forwarding, connection, latency and filtering. In addition to
defining the tests, this document also describes specific formats
for reporting the results of the tests.

A previous document, "Benchmarking Terminology for Firewall
Performance" [1], defines many of the terms that are used in this
document. The terminology document SHOULD be consulted before
attempting to make use of this document.

2. Requirements

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

3. Scope

Firewalls can provide a single point of defense between networks.
Usually, a firewall protects private networks from the public or
shared networks to which it is connected. A firewall can be as
simple as a device that filters different packets or as complex
as a group of devices that combine packet filtering and
application-level proxy or network translation services. This RFC
will focus on developing benchmark testing of DUT/SUTs, wherever
possible, independent of their implementation.

4. Test Setup

Test configurations defined in this document will be confined to
dual-homed and tri-homed as shown in figure 1 and figure 2
respectively.

Firewalls employing dual-homed configurations connect two networks.

One interface of the firewall is attached to the unprotected
network, typically the public network(Internet). The other interface
is connected to the protected network, typically the internal LAN.

In the case of dual-homed configurations, servers which are made
accessible to the public(Unprotected) network are attached to the
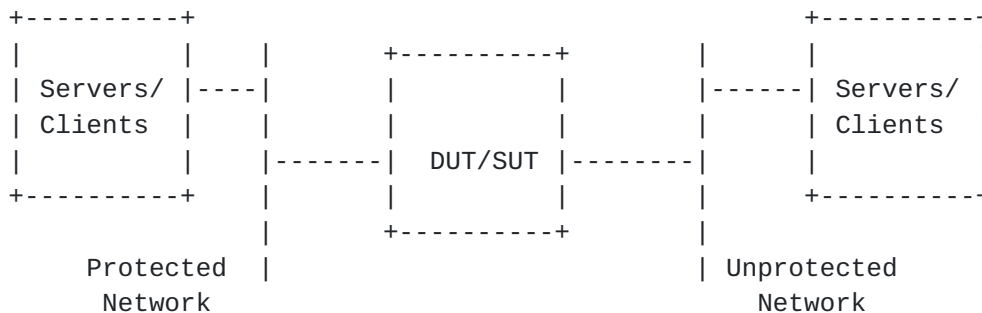private(Protected) network.

```
    +----------+                                      +----------+
    |          |    |        +----------+      |      |          |
    | Servers/ |----|        |          |      |------| Servers/ |
    | Clients  |    |        |          |      |      | Clients  |
    |          |    |--------|  DUT/SUT |--------|     |          |
    +----------+    |        |          |      |      +----------+
                    |        +----------+      |
         Protected  |                          | Unprotected
           Network                               Network
                    Figure 1(Dual-Homed)
```

Tri-homed[1] configurations employ a third segment called a
Demilitarized Zone(DMZ). With tri-homed configurations, servers
accessible to the public network are attached to the DMZ. Tri-Homed
configurations offer additional security by separating server(s)
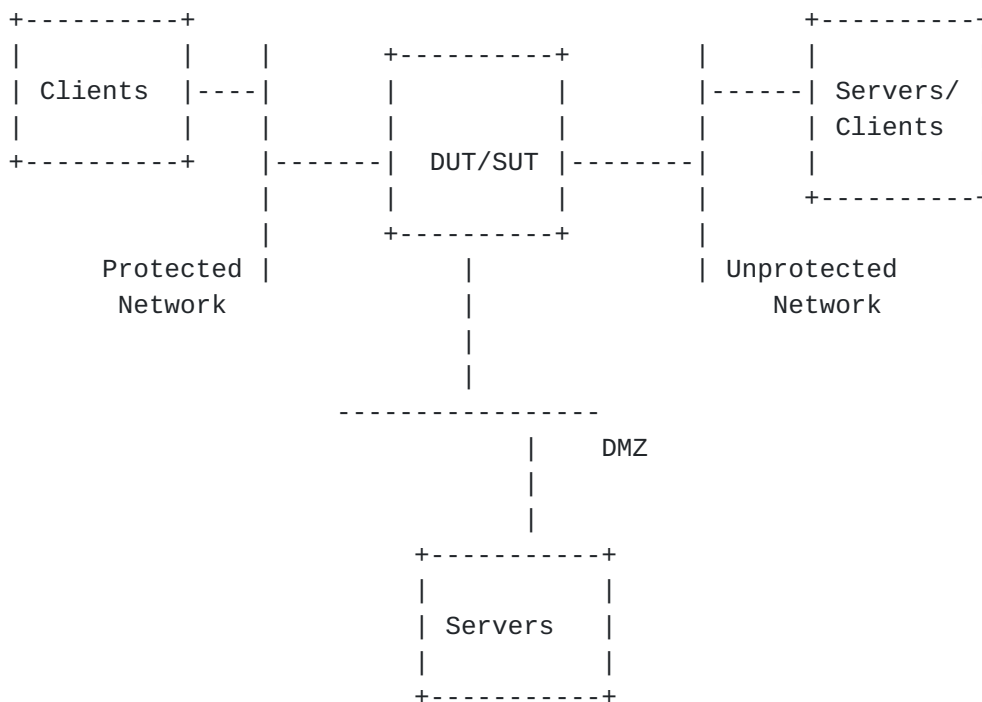accessible to the public network from internal hosts.

```
    +----------+                                      +----------+
    |          |    |        +----------+      |      |          |
    | Clients  |----|        |          |      |------| Servers/ |
    |          |    |        |          |      |      | Clients  |
    +----------+    |--------|  DUT/SUT |--------|     |          |
                    |        |          |      |      +----------+
                    |        +----------+      |
         Protected  |            |             | Unprotected
           Network  |            |               Network
                    |            |
                    |            |
               -----------------
                             |   DMZ
                             |
                             |
                     +-----------+
                     |           |
                     |  Servers  |
                     |           |
                     +-----------+

                    Figure 2(Tri-Homed)
```

[4.1](#) Test Considerations

[4.2](#) Virtual Clients/Servers

Since firewall testing may involve data sources which emulate
multiple users or hosts, the methodology uses the terms virtual
clients/servers. For these firewall tests, virtual clients/servers
specify application layer entities which may not be associated with

   a unique physical interface. For example, four virtual clients may
   originate from the same data source[1]. The test report SHOULD
   indicate the number of virtual clients and virtual servers
   participating in the test.

   Testers MUST synchronize all data sources participating in a test.

4.3 Test Traffic Requirements

   While the function of a firewall is to enforce access control
   policies, the criteria by which those policies are defined vary
   depending on the implementation. Firewalls may use network layer,
   transport layer or, in many cases, application-layer criteria to
   make access-control decisions.

   For the purposes of benchmarking firewall performance this document
   references HTTP 1.1 or higher as the application layer entity,
   although the methodologies may be used as a template for
   benchmarking with other applications. Since testing may involve
   proxy based DUT/SUTs, HTTP version considerations are discussed in
   appendix A.

4.4 DUT/SUT Traffic Flows

   Since the number of interfaces are not fixed, the traffic flows will
   be dependent upon the configuration used in benchmarking the
   DUT/SUT. Note that the term "traffic flows" is associated with
   client-to-server requests.

   For Dual-Homed configurations, there are two unique traffic flows:

      Client   Server
      ------   ------
      Protected   -> Unprotected
      Unprotected -> Protected

   For Tri-Homed configurations, there are three unique traffic flows:

      Client   Server
      ------   ------
      Protected ->   Unprotected
      Protected ->   DMZ
      Unprotected -> DMZ

4.5 Multiple Client/Server Testing

   One or more clients may target multiple servers for a given
   application. Each virtual client MUST initiate connections in a
   round-robin fashion. For example, if the test consisted of six
   virtual clients targeting three servers, the pattern would be as

follows:

```
     Client             Target Server(In order of request)
     #1              1    2    3     1...
       #2                 2    3    1     2...
       #3                 3    1    2     3...
       #4                 1    2    3     1...
       #5                 2    3    1     2...
       #6                 3    1    2     3...
```

4.6 Network Address Translation(NAT)

Many firewalls implement network address translation(NAT), a
function which translates internal host IP addresses attached to
the protected network to a virtual IP address for communicating
across the unprotected network(Internet). This involves additional
processing on the part of the DUT/SUT and may impact performance.
Therefore, tests SHOULD be ran with NAT disabled and NAT enabled
to determine the performance differentials. The test report MUST
indicate whether NAT was enabled or disabled.

4.7 Rule Sets

Rule sets[1] are a collection of access control policies that
determine which packets the DUT/SUT will forward and which it will
reject[1]. Since criteria by which these access control policies may
be defined will vary depending on the capabilities of the DUT/SUT,
the following is limited to providing guidelines for configuring
rule sets when benchmarking the performance of the DUT/SUT.

It is RECOMMENDED that a rule be entered for each host(Virtual
client). In addition, testing SHOULD be performed using different
size rule sets to determine its impact on the performance of the
DUT/SUT. Rule sets MUST be configured in a manner, such that, rules
associated with actual test traffic are configured at the end of the
rule set and not the beginning.

The DUT/SUT SHOULD be configured to deny access to all traffic
which was not previously defined in the rule set. The test report
SHOULD include the DUT/SUT configured rule set(s).

4.7 Web Caching

Some firewalls include caching agents to reduce network load. When
making a request through a caching agent, the caching agent attempts
to service the response from its internal memory. The cache itself
saves responses it receives, such as responses for HTTP GET
requests. Testing SHOULD be performed with any caching agents on the
DUT/SUT disabled.

4.8 Authentication

Access control may involve authentication processes such as user,
client or session authentication. Authentication is usually
performed by devices external to the firewall itself, such as an

   authentication server(s) and may add to the latency of the system.
   Any authentication processes MUST be included as part of connection
   setup process.

5. Benchmarking Tests

5.1 IP Throughput

5.1.1 Objective

   To determine the throughput of network-layer data transversing
   the DUT/SUT, as defined in RFC1242[1]. Note that while RFC1242
   uses the term frames, which is associated with the link layer, the
   procedure uses the term packets, since it is referencing the
   network layer. This test is intended to baseline the ability of
   the DUT/SUT to forward packets at the network layer.

5.1.2 Setup Parameters

   The following parameters MUST be defined:

      Packet size - Number of bytes in the IP packet, exclusive of any
      link layer header or checksums.

      Test Duration - Duration of the test, expressed in seconds.

5.1.3 Procedure

   The tester will offer client/server traffic to the DUT/SUT,
   consisting of unicast IP packets. The tester MUST offer the packets
   at a constant rate. The test MAY consist of either bi-directional or
   unidirectional traffic, with the client offering a unicast stream of
   packets to the server for the latter.

   The test MAY employ an iterative search algorithm. Each iteration
   will involve the tester varying the intended load until the maximum
   rate, at which no packet loss occurs, is found. Since backpressure
   mechanisms may be employed, resulting in the intended load and
   offered load being different, the test SHOULD be performed in either
   a packet based or time based manner as described in RFC2889[7]. As
   with RFC1242, the term packet is used in place of frame. The
   duration of the test portion of each trial MUST be at least 30
   seconds.

   When comparing DUT/SUTs with different MTUs, it is RECOMMENDED to
   limit the maximum IP size tested to the maximum MTU supported by all
   of the DUT/SUTs.

5.1.4 Measurement

[5.1.4.1](#) Network Layer

   Throughput - Maximum offered load, expressed in either bits per

second or packets per second, at which no packet loss is detected.

Forwarding Rate - Forwarding rate, expressed in either bits per second or packets per second, the device is observed to successfully forward to the correct destination interface in response to a specified offered load.

5.1.4 Reporting Format

The test report MUST note the packet size(s), test duration, throughput and forwarding rate. If the test involved offering packets which target more than one segment(Protected, Unprotected or DMZ), the report MUST identify the results as an aggregate throughput measurement.

The throughput results SHOULD be reported in the format of a table with a row for each of the tested packet sizes. There SHOULD be columns for the packet size, the intended load, the offered load, resultant throughput and forwarding rate for each test.

A log file MAY be generated which includes the packet size, test duration and for each iteration:

     - Step Iteration
       - Pass/Fail Status
     - Total packets offered
     - Total packets forwarded
     - Intended load
     - Offered load(If applicable)
     - Forwarding rate

5.2 Concurrent TCP Connection Capacity

5.2.1 Objective

To determine the maximum number of concurrent TCP connections supported through or with the DUT/SUT, as defined in RFC2647[1].

5.2.2 Setup Parameters

The following parameters MUST be defined for all tests:

5.2.2.1 Transport-Layer Setup Parameters

Connection Attempt Rate - The aggregate rate, expressed in connections per second, at which new TCP connection requests are attempted. The rate SHOULD be set at or lower than the maximum rate at which the DUT/SUT can accept connection requests.

Age Time - The time, expressed in seconds, the DUT/SUT will keep a

connection in its connection table after receiving a TCP FIN or RST
packet.

5.2.2.2 Transport-Layer Setup Parameters

Validation Method - HTTP 1.1 or higher MUST be used for this test.

Object Size - Defines the number of bytes, excluding any bytes associated with the HTTP header, to be transferred in response to an HTTP 1.1 or higher GET request.

5.2.3 Procedure

An iterative search algorithm MAY be used to determine the maximum number of concurrent TCP connections supported through or with the DUT/SUT.

For each iteration, the aggregate number of concurrent TCP connections attempted by the virtual client(s) will be varied. The destination address will be that of the server or that of the NAT proxy. The aggregate rate will be defined by connection attempt rate, and will be attempted in a round-robin fashion(See 4.5).

To validate all connections, the virtual client(s) MUST request an object using an HTTP 1.1 or higher GET request. The requests MUST be initiated on each connection after all of the TCP connections have been established.

When testing proxy-based DUT/SUTs, the virtual client(s) MUST request two objects using HTTP 1.1 or higher GET requests. The first GET request is required for connection time establishment measurements as specified in appendix B. The second request is used for validation as previously mentioned. When comparing proxy and non-proxy based DUT/SUTs, the test MUST be performed in the same manner.

Between each iteration, it is RECOMMENDED that the tester issue a TCP RST referencing all connections attempted for the previous iteration, regardless of whether or not the connection attempt was successful. The tester will wait for age time before continuing to the next iteration.

5.2.4 Measurements

5.2.4.1 Application-Layer measurements

Number of objects requested

Number of objects returned

5.2.4.2 Transport-Layer measurements

Maximum concurrent connections - Total number of TCP connections

open for the last successful iteration performed in the search
algorithm.

The following measurements SHOULD be performed on a per iteration
basis:

Minimum connection establishment time - Lowest TCP connection
establishment time measured as defined in appendix B.

Maximum connection establishment time - Highest TCP connection
establishment time measured as defined in appendix B.

Average connection establishment time - The mean of all measurements
of connection establishment times.

Aggregate connection establishment time - The total of all
measurements of connection establishment times.

5.2.5 **Reporting Format**

5.2.5.1 Application-Layer Reporting:

The test report MUST note the object size, number of completed
requests and number of completed responses.

The intermediate results of the search algorithm MAY be reported
in a table format with a column for each iteration. There SHOULD be
rows for the number of requests attempted, number of requests
completed, number of responses attempted and number of responses
completed. The table MAY be combined with the transport-layer
reporting, provided that the table identify this as an application
layer measurement.

Version information:

The test report MUST note the version of HTTP client(s) and
server(s).

5.2.5.2 **Transport-Layer Reporting:**

The test report MUST note the connection attempt rate, age time and
maximum concurrent connections measured.

The intermediate results of the search algorithm MAY be reported
in the format of a table with a column for each iteration. There
SHOULD be rows for the total number of TCP connections attempted,
total number of TCP connections completed, minimum TCP connection
establishment time, maximum TCP connection establishment time,
average connection establishment time and the aggregate connection
establishment time.

5.3 Maximum TCP Connection Establishment Rate

5.3.1 Objective

   To determine the maximum TCP connection establishment rate through
   or with the DUT/SUT, as defined by RFC2647[1].

5.3.2 Setup Parameters

   The following parameters MUST be defined for all tests:

   5.3.2.1 Transport-Layer Setup Parameters

   Number of Connections - Defines the aggregate number of TCP
   connections that must be established.

   Age Time - The time, expressed in seconds, the DUT/SUT will keep a
   connection in it's state table after receiving a TCP FIN or RST
   packet.

   5.3.2.2 Transport-Layer Setup Parameters

   Validation Method - HTTP 1.1 or higher MUST be used for this test.

   Object Size - Defines the number of bytes, excluding any bytes
   associated with the HTTP header, to be transferred in response to an
   HTTP 1.1 or higher GET request.

5.3.3 Procedure Test

   An iterative search algorithm MAY be used to determine the maximum
   rate at which the DUT/SUT can accept TCP connection requests.

   For each iteration, the aggregate rate at which TCP connection
   requests are attempted by the virtual client(s) will be varied. The
   destination address will be that of the server or that of the NAT
   proxy. The aggregate number of connections, defined by number of
   connections, will be attempted in a round-robin fashion(See 4.5).

   The same application-layer object transfers required for validation
   and establishment time measurements as described in the concurrent
   TCP connection capacity test MUST be performed.

   Between each iteration, it is RECOMMENDED that the tester issue a
   TCP RST referencing all connections attempted for the previous
   iteration, regardless of whether or not the connection attempt was
   successful. The tester will wait for age time before continuing to
   the next iteration.

5.3.4 **Measurements**

5.3.4.1 Application-Layer measurements

Number of objects requested

Number of objects returned

5.3.4.2 Transport-Layer measurements

Highest connection rate - Highest rate, in connections per second,
for which for the search algorithm passed.

The following measurements SHOULD performed on a per iteration
basis:

Minimum connection establishment time - Lowest TCP connection
establishment time measured as defined in appendix B.

Maximum connection establishment time - Highest TCP connection
establishment time measured as defined in appendix B.

Average connection establishment time - The mean of all measurements
of connection establishment times.

Aggregate connection establishment time - The total of all
measurements of connection establishment times.

5.3.5 **Reporting Format**

5.3.5.1 Application-Layer Reporting:

The test report MUST note object size(s), number of completed
requests and number of completed responses.

The intermediate results of the search algorithm MAY be reported
in a table format with a column for each iteration. There SHOULD be
rows for the number of requests and responses completed. The table
MAY be combined with the transport-layer reporting, provided that
the table identify this as an application layer measurement.

Version information:

The test report MUST note the version of HTTP client(s) and server(s).

5.3.5.2 Transport-Layer Reporting:

The test report MUST note the number of connections, age time and
highest connection rate measured.

The intermediate results of the search algorithm MAY be reported

in the format of a table with a column for each iteration. There SHOULD be rows for the connection attempt rate, total number of

   TCP connections attempted, total number of TCP connections
   completed, minimum TCP connection establishment time, maximum TCP
   connection establishment time, average connection establishment time
   and the aggregate connection establishment time.

5.4 Maximum TCP Connection Tear Down Rate

5.4.1 Objective

   To determine the maximum TCP connection tear down rate through or
   with the DUT/SUT, as defined by RFC2647[1].

5.4.2 Setup Parameters

   Number of Connections - Defines the number of TCP connections that
   the tester will attempt to tear down.

   Age Time - The time, expressed in seconds, the DUT/SUT will keep a
   connection in it's state table after receiving a TCP FIN or RST
   packet.

5.4.3 Procedure

   An iterative search algorithm MAY be used to determine the maximum
   TCP connection tear down rate. The test iterates through different
   TCP connection tear down rates with a fixed number of TCP
   connections.

   The virtual client(s) will initialize the test by establishing TCP
   connections defined by number of connections. The virtual client(s)
   will then attempt to tear down all of TCP connections, at a rate
   defined by tear down attempt rate. For benchmarking purposes, the
   tester MUST use a TCP FIN when initiating the connection tear down.

   In the case of proxy based DUT/SUTs, the DUT/SUT will itself receive
   the final ACK in the three-way handshake when a connection is being
   torn down. For validation purposes, the virtual client(s) MAY
   verify that the DUT/SUT received the final ACK in the connection tear
   down exchange for all connections by transmitting a TCP datagram
   referencing the previously town down connection. A TCP RST should be
   received in response to the TCP datagram.

5.4.4 Measurements

   Highest connection tear down rate - Highest rate, in connections per
   second, for which all TCP connections were successfully torn down.

   Minimum connection tear down time - Lowest TCP connection tear down
   time measured as defined in appendix C.

Maximum connection tear down time - Highest TCP connection tear down
time measured as defined in [appendix C](#).

   Average connection tear down time - The mean of all measurements of
   connection tear down times.

   Aggregate connection tear down time - The total of all measurements
   of connection tear down times.

5.4.5 Reporting Format

   The test report MUST note the number of connections, age time and
   highest connection tear down rate measured.

   The intermediate results of the search algorithm SHOULD be reported
   in the format of a table with a column for each iteration. There
   SHOULD be rows for the number of TCP tear downs attempted, number
   of TCP connection tear downs completed, minimum TCP connection tear
   down time, maximum TCP connection tear down time, average TCP
   connection tear down time and the aggregate TCP connection tear down
   time.

5.5 Denial Of Service Handling

5.5.1 Objective

   To determine the effect of a denial of service attack on a DUT/SUT
   TCP connection establishment and/or HTTP transfer rates. The denial
   of service handling test MUST be run after obtaining baseline
   measurements from sections 5.3 and/or 5.6.

   The TCP SYN flood attack exploits TCP's three-way handshake
   mechanism by having an attacking source host generate TCP SYN
   packets with random source addresses towards a victim host, thereby
   consuming that host's resources.

5.5.2 Setup Parameters

   Use the same setup parameters as defined in section 5.2.2 or 5.6.2,
   depending on whether testing against the baseline TCP connection
   setup rate test or HTTP transfer rate test, respectfully.

   In addition, the following setup parameters MUST be defined.

   SYN attack rate - Rate, expressed in packets per second, at which
   the server(s) or NAT proxy address is targeted with TCP SYN packets.

5.5.3 Procedure

   Use the same procedure as defined in section 5.3.3 or 5.6.3,
   depending on whether testing against the baseline TCP connection
   establishment rate or HTTP transfer rate test, respectfully. In
   addition, the tester will generate TCP SYN packets targeting the

server(s) IP address or NAT proxy address at a rate defined by SYN
attack rate.

The tester originating the TCP SYN attack MUST be attached to the
unprotected network. In addition, the tester MUST not respond to the
SYN/ACK packets sent by target server or NAT proxy in response to
the SYN packet.

Some firewalls employ mechanisms to guard against SYN attacks. If
such mechanisms exist on the DUT/SUT, tests SHOULD be run with these
mechanisms enabled to determine how well the DUT/SUT can maintain,
under such attacks, the baseline connection establishment rates and
HTTP transfer rates determined in section 5.3 and section 5.6,
respectively.

5.5.4 Measurements

Perform the same measurements as defined in section 5.3.4 or 5.6.4,
depending on whether testing against the baseline TCP connection
establishment rate test or HTTP transfer rate, respectfully.

In addition, the tester SHOULD track TCP SYN packets associated with
the SYN attack which the DUT/SUT forwards on the protected or DMZ
interface(s).

5.5.5 Reporting Format

The test SHOULD use the same reporting format as described in
section 5.3.5 or 5.6.5, depending on whether testing against the
baseline TCP connection establishment rate test or HTTP transfer rate,
respectfully.

In addition, the report MUST indicate a denial of service handling
test, SYN attack rate, number TCP SYN attack packets transmitted
and the number of TCP SYN attack packets forwarded by the DUT/SUT.
The report MUST indicate whether or not the DUT has any SYN attack
mechanisms enabled.

5.6 HTTP Transfer Rate

5.6.1 Objective

To determine the transfer rate of HTTP requested object transversing
the DUT/SUT.

5.6.2 Setup Parameters

The following parameters MUST be defined for all tests:

5.6.2.1 Transport-Layer Setup Parameters

Number of connections - Defines the aggregate number of connections
attempted. The number SHOULD be a multiple of the number of virtual

clients participating in the test

5.6.2.2 Application-Layer Setup Parameters

Session type - The virtual clients/servers MUST use HTTP 1.1 or
higher.

GET requests per connection - Defines the number of HTTP 1.1 or
higher GET requests attempted per connection.

Object Size - Defines the number of bytes, excluding any bytes
associated with the HTTP header, to be transferred in response to an
HTTP 1.1 or higher GET request.

5.6.3 Procedure

Each HTTP 1.1 or higher client will request one or more objects from
an HTTP 1.1 or higher server using one or more HTTP GET requests.
The aggregate number of connections attempted, defined by number of
connections, MUST be evenly divided among all of the participating
virtual clients.

If the virtual client(s) make multiple HTTP GET requests per
connection, it MUST request the same object size for each GET
request. Multiple iterations of this test SHOULD be ran using
different object sizes.

5.6.4 Measurements

5.6.4.1 Application-Layer measurements

Average Transfer Rate - The average transfer rate of the DUT/SUT
MUST be measured and shall be referenced to the requested object(s).
The measurement will start on transmission of the first bit of the
first requested object and end on transmission of the last bit of
the last requested object. The average transfer rate, in bits per
second, will be calculated using the following formula:

$$TRANSFER\ RATE(bit/s) = \frac{OBJECTS * OBJECTSIZE * 8}{DURATION}$$

OBJECTS - Objects successfully transferred

OBJECTSIZE - Object size in bytes

DURATION - Aggregate transfer time based on aforementioned time
          references.

5.6.4.2 Measurements at or below the Transport-Layer

The tester SHOULD make goodput[1] measurements for connection-

oriented protocols at or below the transport layer. Goodput
measurements MUST only reference the protocols payload, excluding
any of the protocols header. In addition, the tester MUST exclude

any bits associated with the connection establishment, connection
tear down, security associations or connection maintenance.

Since connection-oriented protocols require that data be
acknowledged, the offered load[6] will vary over the duration of the
test. When performing forwarding rate measurements, the tester
should measure the average forwarding rate over the duration of the
test.

5.6.5 Reporting Format

5.6.5.1 Application-Layer reporting

The test report MUST note number of GET requests per connection,
and object size.

The transfer rate results SHOULD be reported in tabular form with
a row for each of the object sizes. There SHOULD be a column for the
object size, the number of completed requests, the number of
completed responses, and the transfer rate results for each test.

Failure analysis:

The test report SHOULD indicate the number and percentage of HTTP
GET request or responses that failed to complete.

Version information:

The test report MUST note the version of HTTP client(s) and
server(s).

5.6.5.2 Transport-Layer and below reporting

The test report MUST note the aggregate number of connections. In
addition, the report MUST identify the layer/protocol for which the
measurement was made.

The results SHOULD be in tabular form with a column for each
iteration of the test. There should be columns for transmitted bits,
retransmitted bits and the measured goodput.

Failure analysis:

The test report SHOULD indicate the number and percentage of
connections that failed to complete.

5.7 HTTP Concurrent Transaction Capacity

5.7.1 Objective

   Determine the maximum number of concurrent or simultaneous HTTP
   transactions the DUT/SUT can support. This test is intended to
   find the maximum number of users that can simultaneously access
   web objects.

5.7.2 Setup Parameters

   GET request rate - The aggregate rate, expressed in request per
   second, at which HTTP 1.1 or higher GET requests are offered by the
   virtual client(s).

   Session type - The virtual clients/servers MUST use HTTP 1.1 or
   higher.

5.7.3 Procedure

   An iterative search algorithm MAY be used to determine the maximum
   HTTP concurrent transaction capacity.

   For each iteration, the virtual client(s) will vary the number of
   concurrent or simultaneous HTTP transactions - that is, on-going
   GET requests. The HTTP 1.1 or higher virtual client(s) will request
   one object, across each connection, from an HTTP 1.1 or higher
   server using one HTTP GET request. The aggregate rate at which the
   virtual client(s) will offer the requests will be defined by GET
   request rate.

   The object size requested MUST be large enough, such that, the
   transaction - that is, the request/response cycle -- will exist for
   the duration of the test. At the end of each iteration, the tester
   MUST validate that all transactions are still active. After all of
   the transactions are checked, the transactions MAY be aborted.

5.7.4 Measurements

   Maximum concurrent transactions - Total number of concurrent HTTP
   transactions active for the last successful iteration performed in
   the search algorithm.

5.7.5 Reporting Format

   5.7.5.1 Application-Layer reporting

   The test report MUST note the GET request rate and the maximum
   concurrent transactions measured.

The intermediate results of the search algorithm MAY be reported
in a table format with a column for each iteration. There SHOULD be
rows for the number of concurrent transactions attempted, GET

request rate, number of aborted transactions and number of
transactions active at the end of the test iteration.

Version information:

The test report MUST note the version of HTTP client(s) and
server(s).

5.8 HTTP Transaction Rate

5.8.1 Objective

Determine the maximum HTTP transaction rate that a DUT/SUT can
sustain.

5.8.2 Setup Parameters

Session Type - HTTP 1.1 or higher MUST be used for this test.

Test Duration - Time, expressed in seconds, for which the
virtual client(s) will sustain the attempted GET request rate.
It is RECOMMENDED that the duration be at least 30 seconds.

Requests per connection - Number of object requests per connection.

Object Size - Defines the number of bytes, excluding any bytes
associated with the HTTP header, to be transferred in response to an
HTTP 1.1 or higher GET request.

5.8.3 Procedure

An iterative search algorithm MAY be used to determine the maximum
transaction rate that the DUT/SUT can sustain.

For each iteration, HTTP 1.1 or higher virtual client(s) will
vary the aggregate GET request rate offered to HTTP 1.1 or higher
server(s). The virtual client(s) will maintain the offered request
rate for the defined test duration.

If the tester makes multiple HTTP GET requests per connection, it
MUST request the same object size for each GET request rate.
Multiple iterations of this test MAY be performed with objects of
different sizes.

5.8.4 Measurements

Maximum Transaction Rate - The maximum rate at which all
transactions -- that is all requests/responses cycles -- are
completed.

Transaction Time - The tester SHOULD measure minimum, maximum and average transaction times. The transaction time will start when the virtual client issues the GET request and end when the requesting

   virtual client receives the last bit of the requested object.

5.8.5 Reporting Format

   The test report MUST note the test duration, object size, requests
   per connection and the measured maximum transaction rate.

   The intermediate results of the search algorithm MAY be reported
   in a table format with a column for each iteration. There SHOULD be
   rows for the GET request attempt rate, number of requests attempted,
   number and percentage of requests completed, number of responses
   attempted, number and percentage of responses completed, minimum
   transaction time, average transaction time and maximum transaction
   time.

   Version information:

   The test report MUST note the version of HTTP client(s) and
   server(s).

5.9 Illegal Traffic Handling

 5.9.1 Objective

   To determine the behavior of the DUT/SUT when presented with a
   combination of both legal and Illegal traffic flows. Note that
   Illegal traffic does not refer to an attack, but traffic which
   has been explicitly defined by a rule(s) to drop.

5.9.2 Setup Parameters

   Setup parameters will use the same parameters as specified in the
   HTTP transfer rate test. In addition, the following setup
   parameters MUST be defined:

   Illegal traffic percentage - Percentage of HTTP 1.1 or higher
   connections which have been explicitly defined in a rule(s) to drop.

5.9.3 Procedure

   Each HTTP 1.1 or higher client will request one or more objects from
   an HTTP 1.1 or higher server using one or more HTTP GET requests.
   The aggregate number of connections attempted, defined by number of
   connections, MUST be evenly divided among all of the participating
   virtual clients.

   The virtual client(s) MUST offer the connection requests, both legal
   and illegal, in an evenly distributed manner. Many firewalls have
   the capability to filter on different traffic criteria(IP
   addresses, Port numbers, etc). Testers may run multiple

iterations of this test with the DUT/SUT configured to filter
on different traffic criteria.

5.9.4 **Measurements**

   Tester SHOULD perform the same measurements as defined in HTTP
   test(Section 5.6.4). Unlike the HTTP transfer rate test, the
   tester MUST not include any bits which are associated with illegal
   traffic in its forwarding rate measurements.

5.9.5 **Reporting Format**

   Test report SHOULD be the same as specified in the HTTP
   test(Section 5.6.5).

   In addition, the report MUST note the percentage of illegal HTTP
   connections.

   Failure analysis:

   Test report MUST note the number and percentage of illegal
   connections that were allowed by the DUT/SUT.

5.10 **IP Fragmentation Handling**

5.10.1 **Objective**

   To determine the performance impact when the DUT/SUT is presented
   with IP fragmented[5] traffic. IP packets which have been
   fragmented, due to crossing a network that supports a smaller
   MTU(Maximum Transmission Unit) than the actual IP packet, may
   require the firewall to perform re-assembly prior to the rule set
   being applied.

   While IP fragmentation is a common form of attack, either on the
   firewall itself or on internal hosts, this test will focus on
   determining how the additional processing associated with the
   re-assembly of the packets have on the forwarding rate of the
   DUT/SUT. RFC 1858 addresses some fragmentation attacks that
   get around IP filtering processes used in routers and hosts.

5.10.2 **Setup Parameters**

   The following parameters MUST be defined.

   5.10.2.1 Non-Fragmented Traffic Parameters

   Setup parameters will be the same as defined in the HTTP transfer
   rate test(Sections 5.6.2.1 and 5.6.2.2).

5.10.2.2 **Fragmented Traffic Parameters**

   Packet size - Number of bytes in the IP/UDP packet, exclusive of

link-layer headers and checksums, prior to fragmentation.

MTU - Maximum transmission unit, expressed in bytes. For testing
purposes, this MAY be configured to values smaller than the MTU
supported by the link layer.

Intended Load - Intended load, expressed as percentage of media
utilization.

5.10.3 Procedure

Each HTTP 1.1 or higher client will request one or more objects from
an HTTP 1.1 or higher server using one or more HTTP GET requests.
The aggregate number of connections attempted, defined by number of
connections, MUST be evenly divided among all of the participating
virtual clients. If the virtual client(s) make multiple HTTP GET
requests per connection, it MUST request the same object size for
each GET request.

A tester attached to the unprotected side of the network, will offer
a unidirectional stream of unicast IP/UDP targeting a server
attached to either the protected or DMZ. The tester MUST offer the
unidirectional stream over the duration of the test.

Baseline measurements SHOULD be performed with IP filtering deny
rule(s) to filter fragmented traffic. If the DUT/SUT has logging
capability, the log SHOULD be checked to determine if it contains
the correct information regarding the fragmented traffic.

The test SHOULD be repeated with the DUT/SUT rule set changed to
allow the fragmented traffic through. When running multiple
iterations of the test, it is RECOMMENDED to vary the MTU while
keeping all other parameters constant.

Then setup the DUT/SUT to the policy or rule set the manufacturer
required to be defined to protect against fragmentation attacks and
repeat the measurements outlined in the baseline procedures.

5.10.4 Measurements

Tester SHOULD perform the same measurements as defined in HTTP
test(Section 5.6.4).

Transmitted UDP/IP Packets - Number of UDP packets transmitted by
client.

Received UDP/IP Packets - Number of UDP/IP Packets received by
server.

5.10.5 Reporting Format

5.10.1 Non-Fragmented Traffic

The test report SHOULD be the same as described in [section 5.6.5](#). Note that any forwarding rate measurements for the HTTP traffic

   excludes any bits associated with the fragmented traffic which
   may be forward by the DUT/SUT.

   5.10.2 Fragmented Traffic

   The test report MUST note the packet size, MTU size, intended load,
   number of UDP/IP packets transmitted and number of UDP/IP packets
   forwarded. The test report SHOULD also note whether or not the
   DUT/SUT forwarded the offered UDP/IP traffic fragmented.

5.11 Latency

5.11.1 Objective

   To determine the latency of network-layer or application-layer data
   traversing the DUT/SUT. RFC 1242 [3] defines latency.

5.11.2 Setup Parameters

   The following parameters MUST be defined:

   5.11.2.1 Network-layer Measurements

      Packet size, expressed as the number of bytes in the IP packet,
      exclusive of link-layer headers and checksums.

      Intended load, expressed as percentage of media utilization.

      Test duration, expressed in seconds.

      Test instruments MUST generate packets with unique timestamp
      signatures.

   5.11.2.2 Application-layer Measurements

      Object Size - Defines the number of bytes, excluding any bytes
      associated with the HTTP header, to be transferred in response to
      an HTTP 1.1 or higher GET request. Testers SHOULD use the minimum
      object size supported by the media, but MAY use other object
      sizes as well.

      Connection type. The tester MUST use one HTTP 1.1 or higher
      connection for latency measurements.

      Number of objects requested.

      Number of objects transferred.

      Test duration, expressed in seconds.

Test instruments MUST generate packets with unique timestamp
signatures.

5.11.3 Network-layer procedure

A client will offer a unidirectional stream of unicast packets to a server. The packets MUST use a connectionless protocol like IP or UDP/IP.

The tester MUST offer packets in a steady state. As noted in the latency discussion in RFC 2544 [4], latency measurements MUST be taken at the throughput level -- that is, at the highest offered load with zero packet loss. Measurements taken at the throughput level are the only ones that can legitimately be termed latency.

It is RECOMMENDED that implementers use offered loads not only at the throughput level, but also at load levels that are less than or greater than the throughput level. To avoid confusion with existing terminology, measurements from such tests MUST be labeled as delay rather than latency.

If desired, the tester MAY use a step test in which offered loads increment or decrement through a range of load levels.

The duration of the test portion of each trial MUST be at least 30 seconds.

5.11.4 Application layer procedure

An HTTP 1.1 or higher client will request one or more objects from an HTTP or higher 1.1 server using one or more HTTP GET requests. If the tester makes multiple HTTP GET requests, it MUST request the same-sized object each time. Testers may run multiple iterations of this test with objects of different sizes.

Implementers MAY configure the tester to run for a fixed duration. In this case, the tester MUST report the number of objects requested and returned for the duration of the test. For fixed-duration tests it is RECOMMENDED that the duration be at least 30 seconds.

5.11.5 Measurements

Minimum delay - The smallest delay incurred by data traversing the DUT/SUT at the network layer or application layer, as appropriate.

Maximum delay - The largest delay incurred by data traversing the DUT/SUT at the network layer or application layer, as appropriate.

Average delay - The mean of all measurements of delay incurred by data traversing the DUT/SUT at the network layer or application layer, as appropriate.

Delay distribution - A set of histograms of all delay measurements

observed for data traversing the DUT/SUT at the network layer or
application layer, as appropriate.

5.11.6 Network-layer reporting format

   The test report MUST note the packet size(s), offered load(s) and
   test duration used.

   The latency results SHOULD be reported in the format of a table with
   a row for each of the tested packet sizes. There SHOULD be columns
   for the packet size, the intended rate, the offered rate, and the
   resultant latency or delay values for each test.

5.11.7 Application-layer reporting format

   The test report MUST note the object size(s) and number of requests
   and responses completed. If applicable, the report MUST note the
   test duration if a fixed duration was used.

   The latency results SHOULD be reported in the format of a table with
   a row for each of the object sizes. There SHOULD be columns for the
   object size, the number of completed requests, the number of
   completed responses, and the resultant latency or delay values for
   each test.

   Failure analysis:

   The test report SHOULD indicate the number and percentage of HTTP
   GET request or responses that failed to complete within the test
   duration.

Version information:

   The test report MUST note the version of HTTP client and server.

APPENDIX A: HTTP(HyperText Transfer Protocol)

   The most common versions of HTTP in use today are HTTP/1.0 and
   HTTP/1.1 with the main difference being in regard to persistent
   connections. HTTP 1.0, by default, does not support persistent
   connections. A separate TCP connection is opened up for each
   GET request the client wants to initiate and closed after the
   requested object transfer is completed. While some implementations
   HTTP/1.0 supports persistence through the use of a keep-alive,
   there is no official specification for how the keep-alive operates.
   In addition, HTTP 1.0 proxies do support persistent connection as
   they do not recognize the connection header.

   HTTP/1.1, by default, does support persistent connection and
   is therefore the version that is referenced in this methodology.
   Proxy based DUT/SUTs may monitor the TCP connection and after a
   timeout, close the connection if no activity is detected. The
   duration of this timeout is not defined in the HTTP/1.1
   specification and will vary between DUT/SUTs. If the DUT/SUT
   closes inactive connections, the aging timer on the DUT SHOULD
   be configured for a duration that exceeds the test time.

   While this document cannot foresee future changes to HTTP
   and it impact on the methodologies defined herein, such
   changes should be accommodated for so that newer versions of
   HTTP may be used in benchmarking firewall performance.

APPENDIX B: Connection Establishment Time Measurements

  For purposes of benchmarking firewall performance, the connection
  establishment time will be considered the interval between the
  transmission of the first bit of the first octet of the packet
  carrying the connection request to the DUT/SUT interface to
  receipt of the last bit of the last octet of the last packet of
  the connection setup traffic received on the client or server,
  depending on whether a given connection requires an even or odd
  number of messages, respectfully.

  Some connection oriented protocols, such as TCP, involve an odd
  number of messages when establishing a connection. In the case of
  proxy based DUT/SUTs, the DUT/SUT will terminate the connection,
  setting up a separate connection to the server. Since, in such
  cases, the tester does not own both sides of the connection,
  measurements will be made two different ways. While the following
  describes the measurements with reference to TCP, the methodology
  may be used with other connection oriented protocols which involve
  an odd number of messages.

  For non-proxy based DUT/SUTs , the establishment time shall be

directly measured and is considered to be from the time the first
bit of the first SYN packet is transmitted by the client to the
time the last bit of the final ACK in the three-way handshake is
received by the target server.

If the DUT/SUT is proxy based, the connection establishment time is
considered to be from the time the first bit of the first SYN packet
is transmitted by the client to the time the client transmits the first
bit of the first acknowledged TCP datagram(t4-t0 in the following
timeline).

     t0: Client sends a SYN.
     t1: Proxy sends a SYN/ACK.
     t2: Client sends the final ACK.
     t3: Proxy establishes separate connection with server.
     t4: Client sends TCP datagram to server.
     *t5: Proxy sends ACK of the datagram to client.

   * While t5 is not considered part of the TCP connection establishment,
     acknowledgement of t4 must be received for the connection to be
     considered successful.

APPENDIX C: Connection Tear Time Measurements

  The TCP connection tear down time will be considered the interval
  between the transmission of the first TCP FIN packet transmitted
  by the tester requesting a connection tear down to receipt of the
  ACK packet on the same tester interface.

Appendix D. References

[1] D. Newman, "Benchmarking Terminology for Firewall Devices", RFC 2647,
        August 1999.

[2] R. Fielding, J. Gettys, J. Mogul, H Frystyk, L.Masinter, P. Leach,
    T. Berners-Lee , "Hypertext Transfer Protocol -- HTTP/1.1",
    RFC 2616 June 1999

[3] S. Bradner, editor. "Benchmarking Terminology for Network
    Interconnection Devices," RFC 1242, July 1991.

[4] S. Bradner, J. McQuaid, "Benchmarking Methodology for Network
    Interconnect Devices," RFC 2544, March 1999.

[5] David C. Clark, "IP Datagram Reassembly Algorithm", RFC 815 ,
    July 1982.

[6] Mandeville, R., "Benchmarking Terminology for LAN Switching
    Devices", RFC 2285, February 1998.

[7] Mandeville, R., Perser,J., "Benchmarking Methodology for LAN
    Switching Devices", RFC 2889, August 2000.