Network Working Group   Brooks Hickman
Internet-Draft   Spirent Communications
Expiration Date: December 2001   David Newman
                                                            Network Test
                                                            Terry Martin
                                                        M2networx INC
                                                               June 2001

Benchmarking Methodology for Firewall Performance

<draft-ietf-bmwg-firewall-02.txt>

Status of this Memo

Table of Contents

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 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 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 on a per interface(See 4.1.3) basis.

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. Therefore, the test equipment used to benchmark the DUT/SUT performance MUST consist of real clients and servers generating legitimate layer seven conversations.

For the purposes of benchmarking firewall performance, HTTP 1.1 will be referenced in this document, 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 requests(Connection, object transfers, etc.) 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 NAT(Network Address Translation)

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 SHOULD indicate whether NAT was enabled or disabled.

4.7 Rule Sets

Rule sets[1] are a collection of access control policies that determines which packets the DUT/SUT will forward and which it will reject. The criteria by which these access control policies may be defined will vary depending on the capabilities of the DUT/SUT. The scope of this document is limited to how the rule sets should be applied when testing the DUT/SUT.

The firewall monitors the incoming traffic and checks to make sure that the traffic meets one of the defined rules before allowing it to be forwarded. It is RECOMMENDED that a rule be entered for each host(Virtual client). Although many firewalls permit groups of IP addresses to be defined for a given rule, tests SHOULD be performed with large rule sets, which are more stressful to the DUT/SUT.

The DUT/SUT SHOULD be configured to denies access to all traffic which was not previously defined in the rule set.

4.7 Web Caching

Some firewalls include caching agents in order 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. The report SHOULD indicate whether caching was enabled or disabled on the DUT/SUT.

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 servers 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 Concurrent Connection Capacity

5.1.1 Objective

To determine the maximum number of concurrent connections through or with the DUT/SUT, as defined in RFC2647[1]. This test will employ a step algorithm to obtain the maximum number of concurrent TCP connections that the DUT/SUT can maintain.

5.1.2 Setup Parameters

The following parameters MUST be defined for all tests.

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

Connection Step Count - Defines the number of additional TCP connections attempted for each iteration of the step search algorithm.

Object Size - Defines the number of bytes to be transferred in response to a HTTP 1.1 GET request . It is RECOMMENDED to use the minimum object size supported by the media.

5.1.3 Procedure

Each virtual client will attempt to establish TCP connections to its target server(s), using either the target server's IP address or NAT proxy address, at a fixed rate in a round robin fashion. Each iteration will involve the virtual clients attempting to establish a fixed number of additional TCP connections. This search algorithm will be repeated until either:

- One or more of the additional connection attempts fail to complete.
- One or more of the previously established connections fail.

The test MUST also include application layer data transfers in order to validate the TCP connections since, in the case of proxy based DUT/SUTs, the tester does not own both sides of the connection. For the purposes of validation, the virtual client(s) will request an object from its target server(s) using an HTTP 1.1 GET request, with both the client request and server response excluding the connection-token close in the connection header. In addition, periodic HTTP GET requests MAY be required to keep the underlying TCP connection open(See Appendix A).

5.1.4 Measurements

Maximum concurrent connections - Total number of TCP connections open for the last successful iteration performed in the search algorithm.

5.1.5 Reporting Format

5.1.5.1 Transport-Layer Reporting:

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

5.1.5.2 Application-Layer Reporting:

The test report MUST note the object size(s) and the use of HTTP 1.1 client and server.

5.1.5.3 Log Files

A log file MAY be generated which includes the TCP connection attempt rate, HTTP object size and for each iteration:

- Step Iteration
  - Pass/Fail Status.
- Total TCP connections established.
- Number of previously established TCP connections dropped.
- Number of the additional TCP connections that failed to complete.

5.2 Maximum Connection Setup Rate

5.2.1 Objective

To determine the maximum TCP connection setup rate through or with the DUT/SUT, as defined by RFC2647[1]. This test will employ a search algorithm to obtain the maximum rate at which TCP connections can be established through or with the DUT/SUT.

5.2.2 Setup Parameters

The following parameters MUST be defined.

Initial Attempt Rate - The rate, expressed in connections per second, at which the initial TCP connection requests are attempted.

Number of Connections - Defines the number of TCP connections that must be established. The number MUST be between the number of participating virtual clients and the maximum number supported by the DUT/SUT. It is RECOMMENDED not to exceed the concurrent connection capacity found in section 5.1.

Connection Teardown Rate - The rate, expressed in connections per second, at which the tester will attempt to teardown TCP connections between each iteration. The connection teardown rate SHOULD be set lower than rate at which the DUT/SUT can teardown TCP connections.

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.

Object Size - Defines the number of bytes to be transferred in response to a HTTP 1.1 GET request . It is RECOMMENDED to use the minimum object size supported by the media.

5.2.3 Procedure

An iterative search algorithm will be used to determine the maximum connection rate. This test iterates through different connection rates with a fixed number of connections attempted by the virtual clients to their associated server(s).

Each iteration will use the same connection establishment and connection validation algorithms defined in the concurrent capacity test(See section 5.1).

Between each iteration of the test, the tester must close all connections completed for the previous iteration. In addition, it is RECOMMENDED to abort all unsuccessful connections attempted. The tester will wait for the period of time, specified by age time, before continuing to the next iteration.

5.2.4 Measurements

Highest connection rate - Highest rate, in connections per second, for which all TCP connections completed successfully.

5.2.5 Reporting Format

5.2.5.1 Transport-Layer Reporting:

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

5.1.5.2 Application-Layer Reporting:

The test report MUST note the object size(s) and the use of HTTP 1.1 client and server.
5.1.5.3 Log Files

A log file MAY be generated which includes the total TCP connections attempt, TCP connection teardown rate, age time, HTTP object size and for each iteration:

- Step Iteration
  - Pass/Fail Status.
- Total TCP connections established.
- Number of TCP connections that failed to complete.

5.3 Connection Establishment Time

5.3.1 Objective

To determine the connection establishment times[1] through or with the DUT/SUT as a function of the number of open connections.

A connection for a client/server application is not atomic, in that it not only involves transactions at the application layer, but involves first establishing a connection using one or more underlying connection oriented protocols(TCP, ATM, etc). Therefore, it is encouraged to make separate measurements for each connection oriented protocol required in order to perform the application layer transaction.

5.3.2 Setup Parameters

The following parameters MUST be defined.

Connection Attempt Rate - The rate, expressed in connections per second, at which new TCP connection requests are attempted. It is RECOMMENDED not to exceed the maximum connection rate found in section 5.2.

Connection Attempt Step count - Defines the number of additional TCP connections attempted for each iteration of the step algorithm.

Maximum Attempt Connection Count - Defines the maximum number of TCP connections attempted in the test. It is RECOMMENDED not to exceed the concurrent connection capacity found in section 5.1.

Object Size - Defines the number of bytes to be transferred in response to a HTTP 1.1 GET request.

Number of requests - Defines the number of HTTP 1.1 GET requests per connection. Note that connection, in this case, refers to the underlying transport protocol.

5.3.3 Procedure

Each virtual client will attempt to establish TCP connections to its target server(s) at a fixed rate in a round robin fashion. Each iteration will involve the virtual clients attempting to establish a fixed number of additional connections until the maximum attempt connection count is reached.

As with the concurrent capacity tests, application layer data transfers will be performed. Each virtual client(s) will request one or more objects from its target server(s) using one or more HTTP 1.1 GET request, with both the client request and server response excluding the connection-token close in the connection header. In addition, periodic HTTP GET requests MAY be required to keep the underlying TCP connection open(See appendix A).

Since testing may involve proxy based DUT/SUTs, which terminates the TCP connection, making a direct measurement of the TCP connection establishment time is not possible since the protocol involves an odd number of messages in establishing a connection. Therefore, when testing with proxy based firewalls, the datagram following the final ACK on the three-way handshake will be used in determining the connection setup time.

The following shows the timeline for the TCP connection setup involving a proxy DUT/SUT and is referenced in the measurement section. Note that this method may be applied when measuring other connection oriented protocols involving an odd number of messages in establishing a connection.

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.

5.3.4 Measurements

For each iteration of the test, the tester MUST measure the minimum, maximum and average TCP connection establishment times. Measuring TCP connection establishment times will be made two different ways, depending on whether or not the DUT/SUT is proxy based. If proxy based, the connection establishment time is considered to be from the time the first bit of the SYN packet is transmitted by the client to the time the client transmits the first bit of the TCP datagram, provided that the TCP datagram gets acknowledged(t4-t0 in the above timeline). For DUT/SUTs that are not proxy based, the establishment time shall be directly measured and is considered to be from the time the first bit of the 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.

In addition, the tester SHOULD measure the minimum, maximum and average connection establishment times for all other underlying connection oriented protocols which are required to be established for the client/server application to transfer an object. Each connection oriented protocol has its own set of transactions required for establishing a connection between two hosts or a host and DUT/SUT. 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 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.

5.3.5 Reporting Format

The test report MUST note the TCP connection attempt rate, TCP connection attempt step count and maximum TCP connections attempted, HTTP object size and number of requests per connection.

For each connection oriented protocol the tester measured, the connection establishment time results SHOULD be in tabular form with a row for each iteration of the test. There SHOULD be a column for the iteration count, minimum connection establishment time, average connection establishment time, maximum connection establishment time, attempted connections completed, attempted connections failed.

5.4 Connection Teardown Time

5.4.1 Objective

To determine the connection teardown time[1] through or with the DUT/SUT as a function of the number of open connections. As with the connection establishment time, separate measurements will be taken for each connection oriented protocol involved in closing a connection.

5.4.2 Setup Parameters

The following parameters MUST be defined. Each parameters is configured with the following considerations.

Initial connections - Defines the number of TCP connections to initialize the test with. It is RECOMMENDED not to exceed the concurrent connection capacity found in section 5.1.

Initial connection rate - Defines the rate, in connections per second, at which the initial TCP connections are attempted. It is RECOMMENDED not to exceed the maximum Connection setup rate found in section 5.2.

Teardown attempt rate - The rate at which the tester will attempt to teardown TCP connections.

Teardown step count - Defines the number of TCP connections the tester will attempt to teardown for each iteration of the step algorithm.

Object size - Defines the number of bytes to be transferred across each connection in response to an HTTP 1.1 GET request during the initialization phase of the test as well as periodic GET requests, if required.

5.4.3 Procedure

Prior to beginning a step algorithm, the tester will initialize the test by establishing connections defined by initial connections. The test will use the same algorithm for establishing the connection as described in the connection capacity test(Section 5.1).

For each iteration of the step algorithm, the tester will attempt teardown the number of connections defined by teardown step count at a rate defined by teardown attempt rate. This will be repeated until the tester has attempted to teardown all of the connections.

5.4.4 Measurements

For each iteration of the test, the tester MUST measure the minimum, average and maximum connection teardown times. As with the connection establishment time test, the tester SHOULD measure all connection oriented protocols which are being torn down.

5.4.5 Reporting Format

The test report MUST note the initial connections, initial connection rate, teardown attempt rate, teardown step count and object size.

For each connection oriented protocol the tester measured, the connection teardown time results SHOULD be in tabular form with a row for each iteration of the test. There SHOULD be a column for the iteration count, minimum connection teardown time, average connection teardown time, maximum connection teardown time, attempted teardowns completed, attempted teardown failed.

5.5 Denial Of Service Handling

5.5.1 Objective

To determine the effect of a denial of service attack on a DUT/SUTs connection establishment rates and/or goodput. The Denial Of Service Handling test MUST be run after obtaining baseline measurements from sections 5.2 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.

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 rates and goodput determined in section 5.2 and section 5.6, respectively.

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 connection setup rate test or goodput test, respectfully.

In addition, the following setup parameters MUST be defined.

SYN Attack Rate - Defines the rate, in packets per second at which the server(s) are targeted with TCP SYN packets.

5.5.3 Procedure

Use the same procedure as defined in section 5.2.3 or 5.6.3, depending on whether testing against the baseline connection setup rate test or goodput 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 in response to the SYN packet.

5.5.4 Measurements

Perform the same measurements as defined in section 5.2.4 or 5.6.4, depending on whether testing against the baseline connection setup rate test or goodput test, respectfully.

In addition, the tester SHOULD track 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 5.2.5 or 5.6.5, depending on whether testing against baseline throughput rates or goodput, respectively.

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

5.6 HTTP

5.6.1 Objective

To determine the goodput, as defined by RFC2647, of the DUT/SUT when presented with HTTP traffic flows. The goodput measurement will be based on HTTP objects forwarded to the correct destination interface of the DUT/SUT.

5.6.2 Setup Parameters

The following parameters MUST be defined.

Number of sessions - Defines the number of HTTP 1.1 sessions to be attempted for transferring an HTTP object(s). Number MUST be equal or greater than the number of virtual clients participating in the test. The number SHOULD be a multiple of the virtual clients participating in the test. Note that each session will use one underlying transport layer connection.
Session rate - Defines the rate, in sessions per second, that the HTTP sessions are attempted.

Requests per session - Defines the number of HTTP GET requests per session.

Object Size - Defines the number of bytes to be transferred in response to an HTTP GET request.

5.6.3 HTTP Procedure

Each HTTP 1.1 virtual client will attempt to establish sessions to its HTTP 1.1 target server(s), using either the target server's IP address or NAT proxy address, at a fixed rate in a round robin fashion.

Baseline measurements SHOULD be performed using a single GET request per HTTP session with the minimal object size supported by the media. If the tester makes multiple HTTP GET requests per session, it MUST request the same-sized object each time. Testers may run multiple iterations of this test with objects of different sizes. See appendix A when testing proxy based DUT/SUT regarding HTTP version considerations. 5.6.4 Measurement

Aggregate Goodput - The aggregate bit forwarding rate of the requested HTTP objects. The measurement will start on receipt of the first bit of the first packet containing a requested object which has been successfully transferred and end on receipt of the last packet containing the last requested object that has been successfully transferred. The goodput, in bits per second, can be calculated using the following formula:

$$\text{Goodput} = \frac{\text{OBJECTS} * \text{OBJECTSIZE} * 8}{\text{DURATION}}$$

OBJECTS - Objects successfully transferred

OBJECTSIZE - Object size in bytes

DURATION - Aggregate transfer time based on aforementioned time references.

5.6.5 Reporting Format

The test report MUST note the object size(s), number of sessions, session rate and requests per session.

The goodput results SHOULD be reported in tabular form with a row for each of the object sizes. There SHOULD be columns for the object size, measured goodput and number of successfully transferred objects.

Failure analysis:

The test report SHOULD indicate the number and percentage of HTTP sessions that failed to complete the requested number of transactions, with a transaction being the GET request and successfully returned object.

Version information:

The test report MUST note the use of an HTTP 1.1 client and server.

5.7 IP Fragmentation

5.7.1 Objective

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

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 datagrams has on the goodput of the DUT/SUT.

5.7.2 Setup Parameters

The following parameters MUST be defined.

Trial duration - Trial duration SHOULD be set for 30 seconds.

5.7.2.1 Non-Fragmented Traffic Parameters

Session rate - Defines the rate, in sessions per second, that the HTTP sessions are attempted.

Requests per session - Defines the number of HTTP GET requests per session.

Object Size - Defines the number of bytes to be transferred in response to an HTTP GET request.

5.7.2.1 Fragmented Traffic Parameters

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

Fragmentation Length - Defines the length of the data portion of the IP datagram and MUST be multiple of 8. Testers SHOULD use the minimum value, but MAY use other sizes as well.

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

5.7.3 Procedure

Each HTTP 1.1 virtual client will attempt to establish sessions to its HTTP 1.1 target server(s), using either the target server's IP address or NAT proxy address, at a fixed rate in a round robin fashion. At the same time, a client attached to the unprotected side of the network will offer a unidirectional stream of unicast UDP/IP packets to a server connected to the protected side of the network. The tester MUST offer IP/UDP packets in a steady state.

Baseline measurements SHOULD be performed with a deny rule(s) that filters the 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 fragment length while keeping all other parameters constant.

5.7.4 Measurements

Aggregate Goodput - The aggregate bit forwarding rate of the requested HTTP objects.(See section 5.6). Only objects which have successfully completed transferring within the trial duration are to be included in the goodput measurement.

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

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

5.7.5 Reporting Format

The test report MUST note the test duration.

The test report MUST note the packet size(s), offered load(s) and IP fragmentation length of the UDP/IP traffic. It SHOULD also note whether the DUT/SUT egresses the offered UDP/IP traffic fragmented or not.

The test report MUST note the object size(s), session rate and requests per session.

The results SHOULD be reported in the format of a table with a row for each of the fragmentation lengths. There SHOULD be columns for the fragmentation length, IP/UDP packets transmitted by client, IP/UDP packets received by server, HTTP object size, and measured goodput.

5.8 Illegal Traffic Handling

5.8.1 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 to traffic which has been explicitly defined by a rule(s) to drop.

5.8.2 Setup Parameters

The following parameters MUST be defined.

Number of sessions - Defines the number of HTTP 1.1 sessions to be attempted for transferring an HTTP object(s). Number MUST be equal or greater than the number of virtual clients participating in the test. The number SHOULD be a multiple of the virtual clients participating in the test. Note that each session will use one underlying transport layer connection.
Session rate - Defines the rate, in sessions per second, that the HTTP sessions are attempted.

Requests per session - Defines the number of HTTP GET requests per session.

Object size - Defines the number of bytes to be transferred in response to an HTTP GET request.

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

5.8.3 Procedure

Each HTTP 1.1 virtual client will attempt to establish sessions to its HTTP 1.1 target server(s), using either the target server's IP address or NAT proxy address, at a fixed rate in a round robin fashion.

The tester MUST present 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.

5.8.4 Measurements

Legal sessions allowed - Number and percentage of legal HTTP sessions which completed.

Illegal session allowed - Number and percentage of illegal HTTP session which completed.

5.8.5 Reporting Format

The test report MUST note the number of sessions, session rate, requests per session, percentage of illegal sessions and measurement results. The results SHOULD be reported in the form of a table with a row for each of the object sizes. There SHOULD be columns for the object size, number of legal sessions attempted, number of legal sessions successful, number of illegal sessions attempted and number of illegal sessions successful.

5.9 Latency

5.9.1 Objective

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

5.9.2 Setup Parameters

The following parameters MUST be defined:

5.9.2.1 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.

Offered load, expressed as percentage of media utilization.

Test duration, expressed in seconds.

Test instruments MUST generate packets with unique timestamp signatures.

5.9.2.2 Application-layer Measurements

Object size, expressed as the number of bytes to be transferred across a connection in response to an HTTP 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 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.

5.9.3 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.

5.9.4 Application layer procedure

An HTTP 1.1 client will request one or more objects from an HTTP 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.

5.9.5 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.

5.9.6 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.

5.9.7 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 use of an HTTP 1.1 client and server.

APPENDICES

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. Some implementations of HTTP/1.0 supports persistence by adding an additional header to the request/response:

    Connection: Keep-Alive

However, under HTTP 1.0, 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. When HTTP/1.1 entities want the underlying transport layer connection closed after a transaction has completed, the request/response will include a connection-token close in the connection header:

    Connection: close

If no such connection-token is present, the connection remains open after the transaction is completed. In addition, 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. When performing concurrent connection testing, GET requests MAY need to be issued at a periodic rate so that the proxy does not close the TCP connection.

While this document cannot foresee future changes to HTTP and it's 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. 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.