Network Working Group
Request for Comments: 3511
Category: Informational

B. Hickman
Spirent Communications
D. Newman
Network Test
S. Tadjudin
Spirent Communications
T. Martin
GVNW Consulting Inc

April 2003

Benchmarking Methodology for Firewall Performance

Status of this Memo

Copyright Notice

Abstract

This document discusses and defines a number of tests that may be used to describe the performance characteristics of firewalls. In addition to defining the tests, this document also describes specific formats for reporting the results of the tests.

This document is a product of the Benchmarking Methodology Working Group (BMWG) of the Internet Engineering Task Force (IETF).

Table of Contents

1. Introduction

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

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

In this document, the words that are used to define the significance of each particular requirement are capitalized. These words are:

* "MUST" This word, or the words "REQUIRED" and "SHALL" mean that the item is an absolute requirement of the specification.

* "SHOULD" This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

* "MAY" This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

An implementation is not compliant if it fails to satisfy one or more of the MUST requirements. An implementation that satisfies all the MUST and all the SHOULD requirements is said to be "unconditionally compliant"; one that satisfies all the MUST requirements but not all the SHOULD requirements is said to be "conditionally compliant".

3. Scope

Firewalls can control access between networks. Usually, a firewall protects a private network from public or shared network(s) to which it is connected. A firewall can be as simple as a single device that filters packets or as complex as a group of devices that combine packet filtering and application-level proxy and network translation services. This document focuses on benchmarking firewall performance, wherever possible, independent of 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 [1], typically the public network (Internet). The other interface is connected to the protected network [1], 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.

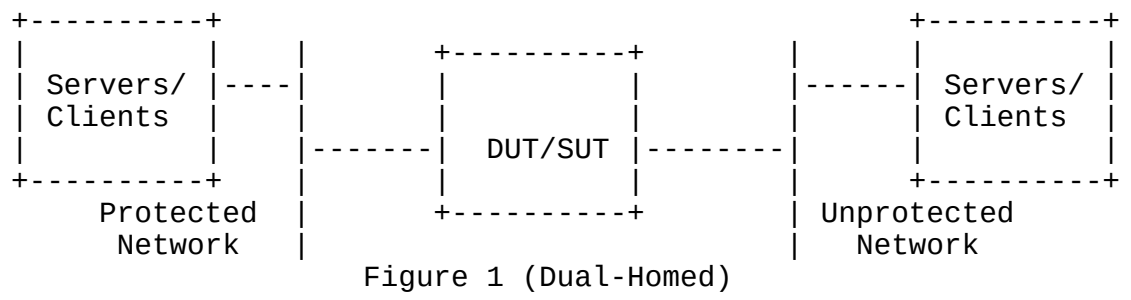

Figure 1 (Dual-Homed)

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.

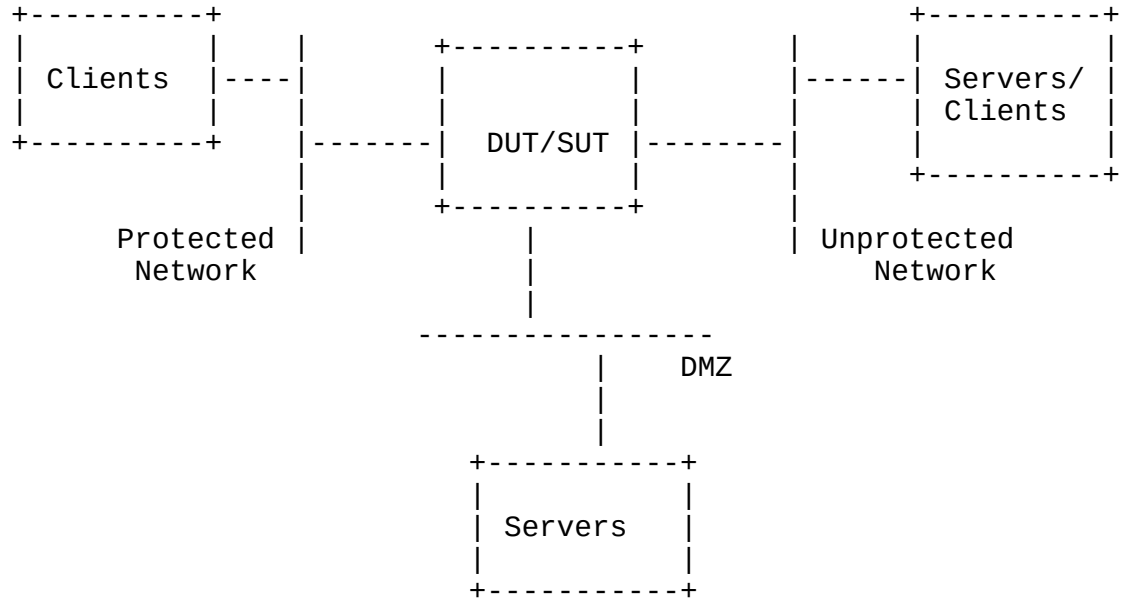

Figure 2 (Tri-Homed)

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 MUST indicate the number of virtual clients and virtual servers participating in the 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. 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) [1], a function which translates private internet addresses to public internet addresses. 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 differential, if any. 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 at 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).

4.8 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.

4.9 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.

4.10 TCP Stack Considerations

Some test instruments allow configuration of one or more TCP stack parameters, thereby influencing the traffic flows which will be offered and impacting performance measurements. While this document does not attempt to specify which TCP parameters should be configurable, any such TCP parameter(s) MUST be noted in the test report. In addition, when comparing multiple DUT/SUTs, the same TCP parameters MUST be used.

5. Benchmarking Tests

5.1 IP Throughput

5.1.1 Objective

To determine the throughput of network-layer data traversing the DUT/SUT, as defined in RFC 1242 [3]. Note that while RFC 1242 uses the term frames, which is associated with the link layer, the procedure uses the term packets, since it is referencing 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 test instrument MUST offer unicast IP packets to the DUT/SUT at a constant rate. The test MAY consist of either bi-directional or unidirectional traffic; for example, an emulated client may offer a unicast stream of packets to an emulated server, or the test instrument may simulate a client/server exchange by offering bidirectional traffic.

This test will employ an iterative search algorithm. Each iteration will involve the test instrument 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 RFC 2889 [5]. As with RFC 1242, the term packet is used in place of frame. The duration of the test portion of each trial MUST be at least 30 seconds.

It is RECOMMENDED to perform the throughput measurements with different packet sizes. When testing with different packet sizes the DUT/SUT configuration MUST remain the same.

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. The bits to be counted are in the IP packet (header plus payload); other fields, such as link-layer headers and trailers, MUST NOT be included in the measurement.

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. The bits to be counted are in the IP packet (header plus payload); other fields, such as link-layer headers and trailers, MUST NOT be included in the measurement.

5.1.5 Reporting Format

The test report MUST note the packet size(s), test duration, throughput and forwarding rate. In addition, the test report MUST conform to the reporting requirements set in section 4, Test Setup. 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.

The intermediate results of the search algorithm MAY be saved in log file 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 RFC 2647 [1]. This test is intended to find the maximum number of entries the DUT/SUT can store in its connection table.

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 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.

Aging 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 Application-Layer Setup Parameters

Validation Method:
HTTP 1.1 or higher MUST be used for this test for both clients and servers. The client and server MUST use the same HTTP version.

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

This test will employ an iterative search algorithm 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 [1] 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 test instrument issue a TCP RST referencing each connection attempted for the previous iteration, regardless of whether or not the connection attempt was successful. The test instrument will wait for aging 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.

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

The test report MUST conform to the reporting requirements set in section 4, Test Setup.

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 tabular format with a column for each iteration. There SHOULD be rows for the number of requests attempted, number and percentage requests completed, number of responses attempted, number and percentage 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, aging time, minimum TCP connection establishment time, maximum TCP connection establishment time, average connection establishment time, aggregate connection establishment 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, number and percentage 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 RFC 2647 [1]. This test is intended to find the maximum rate the DUT/SUT can update its connection table.

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.

Aging 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 Application-Layer Setup Parameters

Validation Method:
HTTP 1.1 or higher MUST be used for this test for both clients and servers. The client and server MUST use the same HTTP version.

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

This test will employ an iterative search algorithm 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 test instrument issue a TCP RST referencing each connection attempted for the previous iteration, regardless of whether or not the connection attempt was successful. The test instrument will wait for aging 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 all connections successfully opened in the search algorithm.

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

The test report MUST conform to the reporting requirements set in section 4, Test Setup.

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 tabular format with a column for each iteration. There SHOULD be rows for the number of requests attempted, number and percentage requests completed, number of responses attempted, number and

percentage 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).

5.3.5.2 Transport-Layer Reporting:

The test report MUST note the number of connections, aging time, minimum TCP connection establishment time, maximum TCP connection establishment time, average connection establishment time, aggregate connection establishment 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 RFC 2647 [1].

5.4.2 Setup Parameters

Number of Connections:
Defines the number of TCP connections that will be attempted to be torn down.

Aging 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.

Close Method:
Defines method for closing TCP connections. The test MUST be performed with either a three-way or four-way handshake. In a four-way handshake, each side sends separate FIN and ACK messages. In a three-way handshake, one side sends a combined FIN/ACK message upon receipt of a FIN.

Close Direction:
Defines whether closing of connections are to be initiated from the client or from the server.

5.4.3 Procedure

This test will employ an iterative search algorithm to determine the maximum TCP connection tear down rate supported by the DUT/SUT. The test iterates through different TCP connection tear down rates with a fixed number of TCP connections.

In the case of proxy based DUT/SUTs, the DUT/SUT will itself receive the ACK in response to issuing a FIN packet to close its side of the TCP connection. For validation purposes, the virtual client or server, whichever is applicable, MAY verify that the DUT/SUT received the final ACK by re-transmitting the final ACK. A TCP RST should be received in response to the retransmitted ACK.

Between each iteration, it is RECOMMENDED that the virtual client(s) or server(s), whichever is applicable, issue a TCP RST referencing each connection which was attempted to be torn down, regardless of whether or not the connection tear down attempt was successful. The test will wait for aging time before continuing to the next iteration.

5.4.4 Measurements

Highest connection tear down rate:
Highest rate, in connections per second, for which all TCP connections were successfully torn down in the search algorithm.

The following tear down time [1] measurements MUST only include connections for which both sides of the connection were successfully torn down. For example, tear down times for connections which are left in a FINWAIT-2 [8] state should not be included:

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, aging time, close method, close direction, 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 and highest connection tear down rate measured. In addition, the test report MUST conform to the reporting requirements set in section 4, Test Setup.

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 number of TCP tear downs attempted, number and percentage of TCP connection tear downs completed, minimum TCP connection tear down time, maximum TCP connection tear down time, average TCP connection tear down time, aggregate TCP connection tear down time and validation failures, if required.

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 5.3.2 or 5.6.2, depending on whether testing against the baseline TCP connection establishment 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 test instrument will generate TCP SYN packets targeting the server(s) IP address or NAT proxy address at a rate defined by SYN attack rate.

The test instrument originating the TCP SYN attack MUST be attached to the unprotected network. In addition, the test instrument 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 and disabled 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 test instrument 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 of 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 traversing 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.

Close Method:
Defines the method for closing TCP connections. The test MUST be performed with either a three-way or four-way handshake. In a four-way handshake, each side sends separate FIN and ACK messages. In a three-way handshake, one side sends a combined FIN/ACK message upon receipt of a FIN.

Close Direction:
Defines whether closing of connections are to be initiated from the client or from the server.

5.6.2.2 Application-Layer Setup Parameters

Session Type:
The virtual clients/servers MUST use HTTP 1.1 or higher. The client and server MUST use the same HTTP version.

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 virtual client will request one or more objects from an HTTP 1.1 or higher server using one or more HTTP GET requests over each connection. 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 may be run with objects of different 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:

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

OBJECTS - Total number of objects successfully transferred across all connections.

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 following measurements SHOULD be performed for each connection-oriented protocol:

Goodput [1]:
Goodput as defined in section 3.17 of RFC 2647. Measurements MUST only reference the protocol payload, excluding any of the protocol header. In addition, the test instrument MUST exclude any bits associated with the connection establishment, connection tear down, security associations [1] or connection maintenance [1].

Since connection-oriented protocols require that data be acknowledged, the offered load [4] will be varying. Therefore, the test instrument should measure the average forwarding rate over the duration of the test. Measurement should start on transmission of the first bit of the payload of the first datagram and end on transmission of the last bit of the payload of the last datagram.

Number of bytes transferred - Total payload bytes transferred.

Number of Timeouts - Total number of timeout events.

Retransmitted bytes - Total number of retransmitted bytes.

5.6.5 Reporting Format

The test report MUST conform to the reporting requirements set in section 4, Test Setup.

5.6.5.1 Application-Layer reporting

The test report MUST note number of GET requests per connection and object size(s).

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

Failure analysis:
The test report SHOULD indicate the number and percentage of HTTP GET request and 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 number of connections, close method, close direction and the protocol for which the measurement was made.

The results SHOULD be reported in tabular form for each of the HTTP object sizes tested. There SHOULD be a row for the total bytes transferred, total timeouts, total retransmitted bytes and and resultant goodput. Note that total bytes refers to total datagram payload bytes transferred. The table MAY be combined with the

application layer reporting, provided the table clearly identifies the protocol for which the measurement was made.

Failure analysis:
The test report SHOULD indicate the number and percentage of connection establishment failures as well as number and percentage of TCP tear down failures.

It is RECOMMENDED that the report include a graph to plot the distribution of both connection establishment failures and connection tear down failures. The x coordinate SHOULD be the elapsed test time, the y coordinate SHOULD be the number of failures for a given sampling period. There SHOULD be two lines on the graph, one for connection failures and one for tear down failures. The graph MUST note the sampling period.

5.7 Maximum HTTP Transaction Rate

5.7.1 Objective

Determine the maximum transaction rate the DUT/SUT can sustain. This test is intended to find the maximum rate at which users can access objects.

5.7.2 Setup Parameters

5.7.2.1 Transport-Layer Setup Parameters

Close Method:
Defines method for closing TCP connections. The test MUST be performed with either a three-way or four-way handshake. In a four-way handshake, each side sends separate FIN and ACK messages. In a three-way handshake, one side sends a combined FIN/ACK message upon receipt of a FIN.

Close Direction:
Defines whether closing of connections are to be initiated from the client or from the server.

5.7.2.2 Application-Layer Setup Parameters

Session Type:
HTTP 1.1 or higher MUST be used for this test. The client and server MUST use the same HTTP version.

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.

5.7.3 Procedure

This test will employ an iterative search algorithm 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 virtual client(s) make multiple HTTP GET requests per connection, it MUST request the same object size for each GET request. Multiple tests MAY be performed with different object sizes.

5.7.4 Measurements

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

Transaction Time:
The test instrument 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.

5.7.5 Reporting Format

The test report MUST conform to the reporting requirements set in section 4, Test Setup.

5.7.5.1 Application-Layer reporting

The test report MUST note the test duration, object size, requests per connection, minimum transaction time, maximum transaction time, average transaction time and maximum transaction rate 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 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).

5.7.5.2 Transport-Layer

The test report MUST note the close method, close direction, number of connections established and number of connections torn down.

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 connections attempted, number and percentage of connections completed, number and percentage of connection tear downs completed. The table MAY be combined with the application layer reporting, provided the table identify this as transport layer measurement.

5.8 Illegal Traffic Handling

5.8.1 Objective

To characterize the behavior of the DUT/SUT when presented with a combination of both legal and Illegal [1] traffic. Note that Illegal traffic does not refer to an attack, but traffic which has been explicitly defined by a rule(s) to drop.

5.8.2 Setup Parameters

Setup parameters will use the same parameters as specified in the HTTP transfer rate test (Section 5.6.2). 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.

5.8.3 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 over each connection. 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.). Multiple iterations of this test MAY be run with the DUT/SUT configured to filter on different traffic criteria.

5.8.4 Measurements

The same measurements as defined in HTTP transfer rate test (Section 5.6.4) SHOULD be performed. Any forwarding rate measurements MUST only include bits which are associated with legal traffic.

5.8.5 Reporting Format

Test reporting format SHOULD be the same as specified in the HTTP transfer rate 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.

5.9 IP Fragmentation Handling

5.9.1 Objective

To determine the performance impact when the DUT/SUT is presented with IP fragmented 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.

5.9.2 Setup Parameters

The following parameters MUST be defined.

5.9.2.1 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).

5.9.2.2 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.

5.9.3 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 over each connection. 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 test instrument attached to the unprotected side of the network, will offer a unidirectional stream of unicast fragmented IP/UDP traffic, targeting a server attached to either the protected or DMZ segment. The test instrument MUST offer the unidirectional stream over the duration of the test, that is, duration over which the HTTP traffic is being offered.

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.

5.9.4 Measurements

Test instrument 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.

5.9.5 Reporting Format

5.9.5.1 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.

5.9.5.2 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.

5.10 Latency

5.10.1 Objective

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

5.10.2 Setup Parameters

The following parameters MUST be defined:

5.10.2.1 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.

The test instruments MUST generate packets with unique timestamp signatures.

5.10.2.2 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. The minimum object size supported by the media SHOULD be used, but other object sizes MAY be used as well.

Connection type:
The test instrument 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.

5.10.3 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 test instrument MUST offer packets in a steady state. As noted in the latency discussion in RFC 2544 [2], 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.

It is RECOMMENDED to perform the latency measurements with different packet sizes. When testing with different packet sizes the DUT/SUT configuration MUST remain the same.

If desired, a step test MAY be used 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.

5.10.4 Application layer procedure

An 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. If the test instrument makes multiple HTTP GET requests, it MUST request the same-sized object each time. Multiple iterations of this test may be performed with objects of different sizes.

Implementers MAY configure the test instrument to run for a fixed duration. In this case, the test instrument 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.

5.10.5 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.

5.10.6 Network-layer reporting format

The test report MUST note the packet size(s), offered load(s) and test duration used. In addition, the test report MUST conform to the reporting requirements set in section 4, Test Setup.

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.

5.10.7 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. In addition, the test report MUST conform to the reporting requirements set in section 4, Test Setup.

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.

6. References

6.1 Normative References

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

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

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

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

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

6.2 Informative References

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

[7] Clark, D., "IP Datagram Reassembly Algorithm", RFC 815, July 1982.

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

7. Security Considerations

The primary goal of this document is to provide methodologies in benchmarking firewall performance. While there is some overlap between performance and security issues, assessment of firewall security is outside the scope of this document.

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

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 test instrument 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.

When testing 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 Down Time Measurements

While TCP connections are full duplex, tearing down of such connections are performed in a simplex fashion, that is, FIN segments are sent by each host/device terminating each side of the TCP connection.

When making connection tear down times measurements, such measurements will be made from the perspective of the entity, that is, virtual client/server initiating the connection tear down request. In addition, the measurement will be performed in the same manner, independent of whether or not the DUT/SUT is proxy-based. The connection tear down will be considered the interval between the transmission of the first bit of the first TCP FIN packet transmitted by the virtual client or server, whichever is applicable, requesting a connection tear down to receipt of the last bit of the corresponding ACK packet on the same virtual client/server interface.

Authors' Addresses

Brooks Hickman
Spirent Communications
26750 Agoura Road
Calabasas, CA 91302
USA

Phone: + 1 818 676 2412
EMail: brooks.hickman@spirentcom.com

David Newman
Network Test Inc.
31324 Via Colinas, Suite 113
Westlake Village, CA 91362-6761
USA

Phone: + 1 818 889-0011
EMail: dnewman@networktest.com

Saldju Tadjudin
Spirent Communications
26750 Agoura Road
Calabasas, CA 91302
USA

Phone: + 1 818 676 2468
EMail: Saldju.Tadjudin@spirentcom.com

Terry Martin
GVNW Consulting Inc.
8050 SW Warm Springs Road
Tualatin Or. 97062
USA

Phone: + 1 503 612 4422
EMail: tmartin@gvnw.com

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.