

Firewall Testing Methodology using Spirent Solutions

Application Note 01

July, 2003



Spirent Communications

26750 Agoura Road
Calabasas Hills, CA
91302 USA
e: enterprise@spirentcom.com

Sales Contacts:

North America

+1 800-927-2660

Europe, Middle East, Africa

+33-1-6137-2250

Asia Pacific

+852-2166-8382

All Other Regions

+1 818-676-2683

www.spirentcom.com/enterprise

Application Note 01: Firewall Testing Methodology using Spirent Solutions
authored by Hemendra Godbole and Norm Conklin.

Copyright 2003 by Spirent Communications, Inc. All rights reserved.

Table of Contents

1. Overview	1
2. Introduction.....	1
3. Inadequate Measurement... A Recipe for Disaster	3
4. Realism-Testing of Your Firewall with Spirent	4
4.1 “Firewall Basic” – Establishing a Reference Baseline Test	5
4.1.1 Objective	5
4.1.2 Requirements.....	5
4.1.3 Operational Parameters to Establish.....	5
4.1.4 Running “Firewall Basic”	5
4.2 “Firewall Stress” – Stress Testing Your Firewall.....	8
4.2.1 Objective	8
4.2.2 Additional Requirements	8
4.2.3 Operational Parameters to Establish.....	8
4.2.4 Running “Firewall Stress”	8
4.4 “Firewall Load” – Load Testing Your Firewall.....	13
4.4.1 Objective.....	13
4.4.2 Additional Requirement	13
4.4.3 Running “Firewall Load”	13
5. Summary	19
6. Background	21

1. Overview

Security and network engineers are faced with complex and continually increasing threats to their network computing resources. The firewall, a critical item in the overall security policy of a corporation, is the first line of defense to unauthorized entry into a corporate network. While firewall providers attempt to simulate the many types of attacks that can cripple a corporate network, it is unrealistic to believe they can do so given the unique operating requirements existing for individual businesses. For this reason, security and network engineers must conduct their own testing based on their intimate knowledge of their corporate network operation to ensure network-operating security and availability.

This application note will assist you in adding your firewall into a Spirent testing framework. You will learn how to create and run basic, stress and load tests and calibrate your firewall, thereby combining the *realism of production networks with the precision of lab testing*. By implementing the tests as suggested in this paper, you can ensure that your firewall meets your security and availability standards. An added benefit is that you will have a comprehensive testing framework for end-to-end, multi-device realism testing. You can therefore extend the tests discussed here to include other devices such as Server Load Balancers (SLB) and Intrusion Detection Systems (IDS) in conjunction with your firewall testing.

Note: This application note assumes that you are familiar with configuring and running basic tests using Spirent's *Avalanche/Reflector* appliances and/or the Terametrics cards in Spirent's *SmartBits* chassis.

2. Introduction

A *firewall* separates an unprotected network, such as the Internet, from a protected one – a corporate network. A corporation defines a set of *security policies* that govern corporate network traffic, both incoming and outgoing that traverses a Firewall. Security policies govern Firewall policies that conditionally allow traffic to pass through the firewall. The firewall policies generally relate to layers 4–7 of the International Standards Organization (ISO) Open Systems Interconnect (OSI) model for networking.

Firewalls use a combination of authentication, packet filtering and application gateway techniques and are one element of an overall security policy. *Authentication* is the process of verifying entities the protected network wishes to connect to (or verifying the entities wishing to connect to it). *Packet filtering* is the process whereby the firewall analyzes incoming or outgoing data against filters (security policies) to determine whether to allow the data to pass. Application gateways exist so as prevent direct network (unprotected) to network (protected) connection, which increases the risk of harm to a network.

The following firewall types reflect the evolution of firewall technology:

- *Packet filter* – analyzes data at the transport layer against the filters that define data flow. It is the least secure firewall type because it works at the transport layer rather than the application layer. Since it knows little about the connection state, it is also referred to as a 'stateless' firewall.
- *Stateful inspection* – permits data flow based on established sessions meeting a defined security policy. Active sessions are maintained in a session table. Packets as-

sociated with sessions not in the table are dropped (not allowed to pass through the firewall). This is a more secure firewall than the packet filter because there is tighter definition on what type of data can pass through the firewall. Because it matches data against policies contained in a database, it not only restricts based on the transport and session layers, but up through the application layer as well. Stateful firewalls mandate detailed Layer 4–7 testing to ensure their applicability.

- *Application proxy* – consists of two TCP connections, one between the packet source and the firewall, the other between the firewall and the packet destination. The proxy performs the gatekeeping on whether to allow incoming packets through to the destination based on the application rules. Each different application requires its own proxy. Since the packet inspection occurs at the networking layer, more overhead is required for an application proxy. Further, a proxy firewall needs to process each protocol passing through it – thereby limiting its scalability. For these reasons, even though an application proxy may provide more specific controls than stateful inspection, *stateful inspection firewalls are more commonly implemented and are the focus of this paper from a testing perspective.*

As stated, stateful firewalls are only one element of an overall enterprise security policy. A firewall needs to interoperate seamlessly with other networking and security devices, potentially from multiple vendors, and are subject to unique enterprise traffic characteristics. It is therefore essential to test a firewall in the context of an enterprise IT infrastructure – beyond vendor claims or 3rd-party labs regarding its applicability.

An interesting trend is that firewalls have matured to become perimeter security platforms that aggregate additional security products such as IPSec VPN gateways and Intrusion Detection Systems (IDS). While these integrated firewalls offer greater manageability and incorporate newer techniques such as deep packet inspection, critical questions remain as you determine if an aggregated firewall platform is right for you:

- Does your aggregated platform perform as desired for the stateful firewall functions?
- Is the platform CPU and memory constrained to the point where, as an inline device serving multiple functions, its saturation throttles your infrastructure availability?
- Does your aggregated firewall evaluation framework also test its scalability? While under realistic loads consisting of multiple protocols at varying rates - *with shared CPU and memory resources typical of most aggregated platforms* – does your aggregated firewall operate at acceptable levels of security and availability?
- By consolidating multiple services, did your aggregate firewall truly simplify your end-to-end security testing? Did it create new testing requirements or complexities? Did your testing framework help with migrating from a distributed to a unified firewall platform?

By combining the realism of production networks with the precision of lab testing, Spirent offers a comprehensive methodology along with detailed testing to help you address these questions. This

ensures your investments in the right firewall to scale with current requirements and affords you the opportunity to evaluate dominant security trends for applicability to your specific environment.

3. Inadequate Measurement... A Recipe for Disaster

Evaluating your firewall is fraught with risk. Risk not just from your architecture – can it sufficiently handle what you want it to do – but also from an increasing sophistication of attacks. How can these risks be mitigated? How do you ensure both security and availability? It is not just a matter of functionally verifying operation and that your CPU and memory are sufficient at some benchmark that likely doesn't mimic your network operation. *Attacks at load* are the most dangerous for security and availability. Is it possible to realistically stress test all relevant firewall parameters including:

- Throughput, including not inhibiting runtime when stopping attacks to ensure 24/7/forever availability
- Number of new connections/second
- Total number of concurrent connections supported by the firewall
- New connections/second *while under load* – maintaining a high number of concurrent connections
- Ability to stop well-known denial-of-service attacks while under load
- Ability to simulate new DDOS attacks
- Number of policies
- Transit latency through firewalls at realistic loads
- Multi-protocol traffic consisting potentially of HTTP, HTTPS, RTP/RTSP, telnet, DNS, and SMTP/POP3 protocols
- Failover in an active-active deployment

Also increasing your risk are the intricacies of *your* operating environment. Neither vendors nor 3rd-party laboratories can recreate your network realism to uncover the load characteristics necessary to determine that you are safe from attacks. Simply extrapolating vendor benchmarks or typical lab testing equates to inadequate measurement and a recipe for disaster. For example, how can one adequately extrapolate the affect that 40 policies have on throughput without specifically knowing when the peaks occur within your network and in what combinations? How can these adequately extrapolate the effect to multi-protocol throughput while denying attacks on one protocol?

Spirent solutions are uniquely positioned to help you evaluate your firewall because:

- Dominant firewall vendors use Spirent products to benchmark their devices. Vendor benchmarks derived from the testing may still not adequately reflect your unique traffic-mix, load and end-to-end security requirements.
- Several 3rd-party labs, including ICSA, Tolly, VeriTest, Neophasis and NSS, use Spirent products to perform firewall certification. Again, this level of testing may not adequately reflect your unique needs.

- By using the same testing solutions used by your firewall vendors and 3rd-party testing labs, you can leverage the test scenarios and configurations to remove any variations added by testing frameworks. You also simplify your testing strategies and gain a greater degree of confidence in selecting the right firewall that suits your unique needs thereby mitigating any new risks added by your testing solutions

The following discussion shows how Spirent enables you to perform a comprehensive firewall evaluation by helping you:

- Conduct a set of basic tests using the Spirent **Avalanche**, **Reflector** and **SmartBits** products with your firewall to ensure that it meets your expectations as a key element of your security policy.
- Conduct a set of advanced tests that ensure the end-to-end behavior of your firewall in conjunction with other devices in your IT infrastructure. These could range from security appliances such as IDS and SSL-accelerator devices, or content-appliances such as SLBs that are individually tested by the **Avalanche/Reflector** prior to being introduced in your network. Spirent recommends a methodical series of tests beginning with the individual firewall and progressing to the other infrastructure elements to reflect end-to-end requirements that closely simulate your real-life production network.
- Evaluate your firewall effectiveness with new business drivers, such as network consolidation and Intrusion Prevention Systems (IPS), for internal attack mitigation.

The details of these tests follow. Readers are assumed to be familiar with the basic configuration, execution and result-interpretation of simple tests on Avalanche and Reflector appliances or the Smartbits WebAvalanche platforms.

4. Realism-Testing of Your Firewall with Spirent

This section provides a basic firewall testing methodology using Spirent's **Avalanche/ Reflector**, or **SmartBits** platforms. Conducting these tests enables you to increasingly reflect your production network realism by adding more parameters to your firewall tests and include additional devices in series with your firewall. This evaluates your firewall in a context similar to your production network, which reduces deployment risk.

The tests discussed here can be run on the **Avalanche/Reflector** or **SmartBits** platforms.

The set of tests described are:

- “Firewall Basic” – characterizes operating limits of the testing framework.
- “Firewall Stress” – measures the operating limits of your firewall.
- “Firewall Load” – measures how well your firewall maintains availability under load.

The test discussion assumes you are familiar with:

- **Avalanche/Reflector** basics such as logging into the **Avalanche/Reflector** appliances (or **SmartBits/WebAvalanche** cards) and running the basic pre-packaged tests (such as “Echo” on **Reflector** and “SPI” on **Avalanche**).
- Firewalls and networking basics (such as IP address allocation and switched networks).

4.1 “Firewall Basic” – Establishing a Reference Baseline Test

4.1.1 Objective

Before characterizing a firewall, the testing framework operating limits must be known. The basic testing framework consists of back-to-back **Avalanche/Reflector** devices connected via a managed switch. The switch is not mandatory; however, it is highly recommended as it simplifies network connectivity when adding a firewall to the testing framework.

Once you establish the testing equipment operating limits, inserting a firewall between the **Avalanche/Reflector** eliminates most headaches associated with the classic test problem: In case of test failure, is it the firewall or the testing framework that is causing the failure?

4.1.2 Requirements

- **Avalanche** and **Reflector** (or **SmartBits/WebAvalanche** cards)
- Managed Layer 2/3 Switch (with greater throughput than that expected from the firewall)
- Management console - any PC with Ethernet connectivity with a browser, JVM and Adobe Acrobat)

4.1.3 Operational Parameters to Establish

- Connections per second (CPS) using HTP 1.0 and FTP
- Maximum number of concurrent connections (HTTP and FTP)

4.1.4 Running “Firewall Basic”

1. Connect your **Avalanche, Reflector**, management console and Layer 3 switch as shown in Figure 1 (next page).
2. Ensure correct IP addressing between all devices:
 - **Avalanche** management address: 192.168.42.2 (default)
 - **Reflector** management address: 192.168.42.3 (default)
 - Management console address: 192.168.42.5 (given for this test)
3. Copy “Echo” on **Reflector** to a new test called “FW Basic Reflector”



Figure 1: Determining Avalanche/Reflector Operating Limits.

- Setup servers to support HTTP 1.0 and FTP (1 KB file size)
- Ensure that the test reaches steady state.

4. Configure “Firewall Basic” test on **Avalanche**.

- Select a pre-configured test (SPI) and copy it as “Firewall Basic” test.
- You can now modify “Firewall Basic” and save it as a new test.
 - Setup client-side traffic to consist of two protocols: HTTP 1.0 and FTP in user-profiles.
 - Configure the *Load Spec* to test the desired limits.

For example, if the firewall is benchmarked at 1,000 new connections/sec and 10,000 concurrent connections – and these are the parameters to be tested – run two separate sequential tests to ensure the testing framework can meet or exceed these benchmarks.

- Using “connections/sec” as the load-spec, run tests to determine that you can run over 1000 new CPS.
 - Using the same load-spec, ensure that you can sustain over 10,000 connections during the “steady state” of the test run.
5. Check network port configurations and run “Firewall Basic” on **Avalanche**.
6. Debug any failures that exist that might be due to improper network configurations or Ethernet connectivity between the switch/devices. Failures could include:

- A leak of disallowed traffic – especially if the test is run over a long interval (48hours) – that might uncover a memory leak
 - Timeouts for new incoming connections.
 - Resets of incoming connections while processing valid traffic.
7. Ensure that the **Avalanche/Reflector** and Layer 3-based test-bed operating limits are higher than those of the firewall being tested.
- Increase the load-spec gradually while maintaining “no failing tests.”
 - Note the operational limits of the test-bed for the specific “Firewall Basic” test (HTTP 1.0 and FTP protocols only).

With no failing tests, you are ready to add a firewall to the test-bed for *stress* and *load* testing, building upon your “Firewall Basic” test by incorporating greater realism to more closely represent your production network.

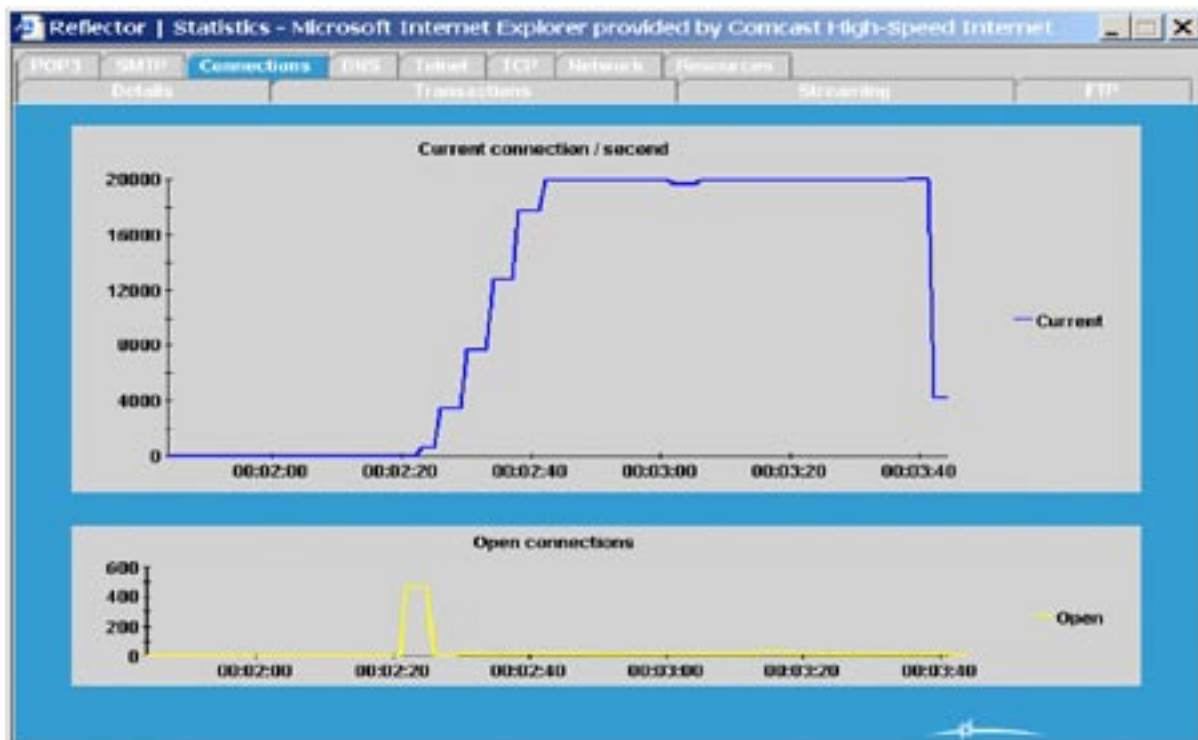


Figure 2: CPS and maximum open connections for a specific test-load.

4.2 “Firewall Stress” – Stress Testing Your Firewall

4.2.1 Objective

Determine your firewall operating limits for parameters such as connections per second and open connections.

4.2.2 Additional Requirements

Your firewall, typically with an *external*, *internal* and DMZ set of ports

Note: Barring the insertion of a firewall into the test-bed, minimal changes are required in the test configuration. In contrast to “Firewall Basic,” which was designed to test the operating limits of the *test equipment*, “Firewall Stress” focuses on the firewall.

4.2.3 Operational Parameters to Establish

- Connections per second (cps) using HTTP 1.0 and FTP
- Maximum number of concurrent connections (htp and FTP)

4.2.4 Running “Firewall Stress”

1. Connect your Firewall into the test-bed as shown in Figure 3.
2. Configure appropriate network IP address:

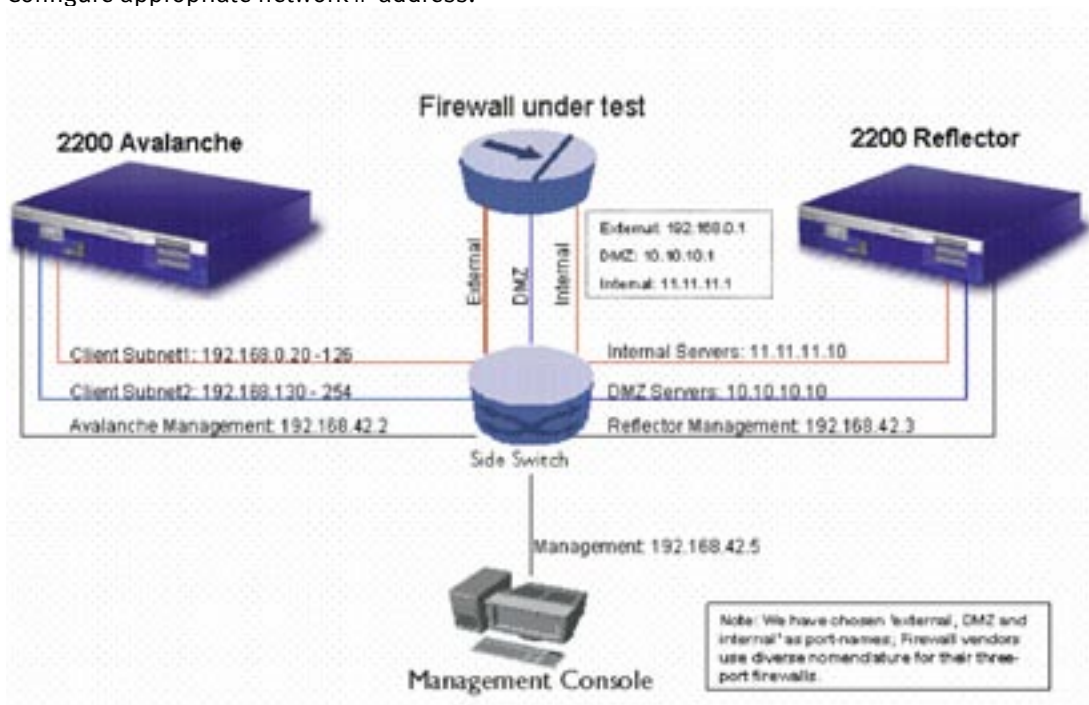


Figure 3: Testing Framework for “Firewall Load” and “Firewall Stress” tests.

- Assign a typical three-port firewall comprising external (unprotected), DMZ (unprotected) and internal (protected) ports on the **Avalanche** and **Reflector** as follows:

a) **Reflector**

- i) DMZ (minimally protected) servers: 10.10.10.10, Virtual Router: 10.10.10.2
- ii) Internal (protected) servers: 11.11.11.10, Virtual Router: 11.11.11.2

b) **Avalanche**

- i) Clients (unprotected): 192.168.0.20-126, Virtual Router: 192.168.0.2
- ii) Clients (unprotected): 192.168.0.130-254, Virtual Router: 192.168.0.129

Two client-banks are needed to create a large number of virtual users in the interface profile on the Avalanche with one-bank recommended as generating traffic for the DMZ servers and the other for the protected servers simulated on the Reflector as in Figure 3.

Firewall IP addressing (data ports):

- a) Internal address (protected): 11.11.11.1
 - b) Inside address (DMZ): 10.10.10.1
 - c) External address (unprotected): 192.168.0.1
 - d) Firewall configuration (over serial port)
- Configure the firewall:

The following is an *example* of the type of firewall configuration needed. It contains network IP addresses and policies, and is implemented with a simple “allow http” and “allow ftp” rule set (change the syntax as per your vendor’s requirements). Note that this is strictly an example to illustrate the testing methodology; each firewall vendor will have different configuration scripts and tools.

```
nameif ethernet0 outside security0
```

```
nameif ethernet1 inside security100
```

```
interface ethernet0 auto
```

```
interface ethernet1 auto
```

```
ip address outside 192.168.0.1 255.255.0.0
```

```
ip address inside 10.10.10.1 255.255.255.0
```

```
arp timeout 14400
```

```
static (inside,outside) 10.10.10.10 10.10.10.10
```

```
static (inside,outside) 10.10.10.11 10.10.10.11
```

```
conduit (inside,outside) 10.10.10.10 80 tcp 0.0.0.0 0.0.0.0
```

```
conduit (inside,outside) 10.10.10.11 21 tcp 0.0.0.0 0.0.0.0
```

```
route outside 192.168.0.128 255.255.255.128 192.168.0.129 1
route outside 192.168.0.0 255.255.255.128 192.168.0.2 1
timeout xlate 24:00:00 conn 12:00:00 udp 0:02:00
timeout rpc 0:10:00 h323 0:05:00 uauth 0:05:00
no snmp-server location
no snmp-server contact
snmp-server community public
mtu outside 1500
mtu inside 1500
```

3. Select and copy “Firewall Basic” to “Firewall Stress CPS”:

- To simplify things, initially reduce the number of IP protocols to only HTTP and add FTP later (step 6).

4. Modify the load spec to calibrate CPS:

- Target 70% of claimed benchmark numbers and ensure that you have successful results (this is a quick safety check for firewall functionality)

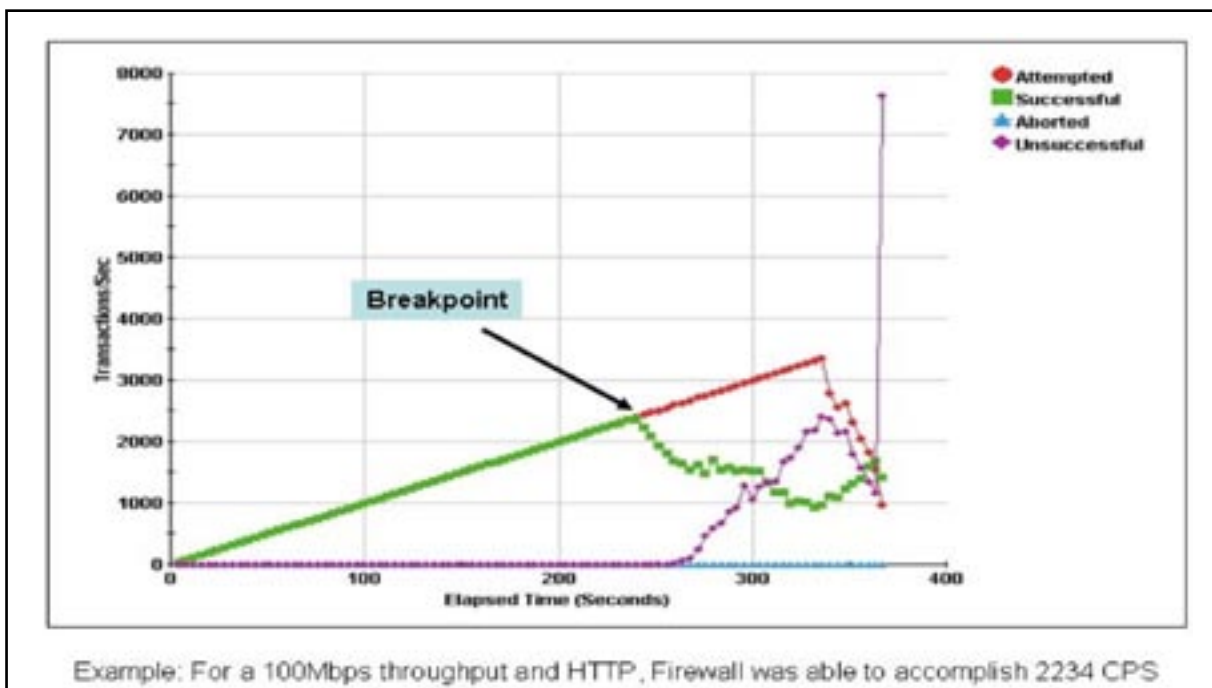


Figure 4: Determining Maximum CPS for Firewall under test.

- Modify load granularity in CPS on the Avalanche load profile at the startup, steady state and teardown state to ensure that your total CPS – reflected as the sum total of each of the test steps (different test states) – exceeds the specified firewall CPS benchmark by a factor of 30-50%.
5. Run “Firewall Stress CPS”
 - Determine firewall failure points by going through the test results files and checking for any of the firewall failure mechanisms mentioned in (5a) of “Firewall Basic Test.” Figure 4 (previous page) depicts an example of the results.
 6. Add FTP to the IP protocol mix and re-run “Firewall Stress” by adding URLs in the user profile on **Avalanche** as well as configuring FTP servers on the **Reflector**.

In contrast to the short-lived HTTP connections, FTP transactions can be long-lived and consume firewall memory resources (connection tables). New connections consume firewall CPU and memory resources. With firewall memory consumed by FTP-related connection tables, you should expect the breakpoint in Figure 4 to shift further left for CPS rates. Figure 5 shows that at times, the drop in CPS is non-intuitive and dramatic.

Note: that with all parameters held constant, just the addition of FTP resulted in a greater than 75% drop in CPS for a sample firewall used for demonstration purposes. Your firewall can have different breakpoints.

Firewall vendors do not necessarily provide these data. Adding more protocols (such as RTSP/ RTP) that require more resources inside the firewall can lead to even more dramatic degradation in firewall performance.

Note: In figure 5, a single URL for HTTP 1.0 and FTP (control plane transactions) amounted to a 1:1 ratio between Connections and Transactions per second (TPS), hence the labels for TPS and CPS for this example are synonymous.

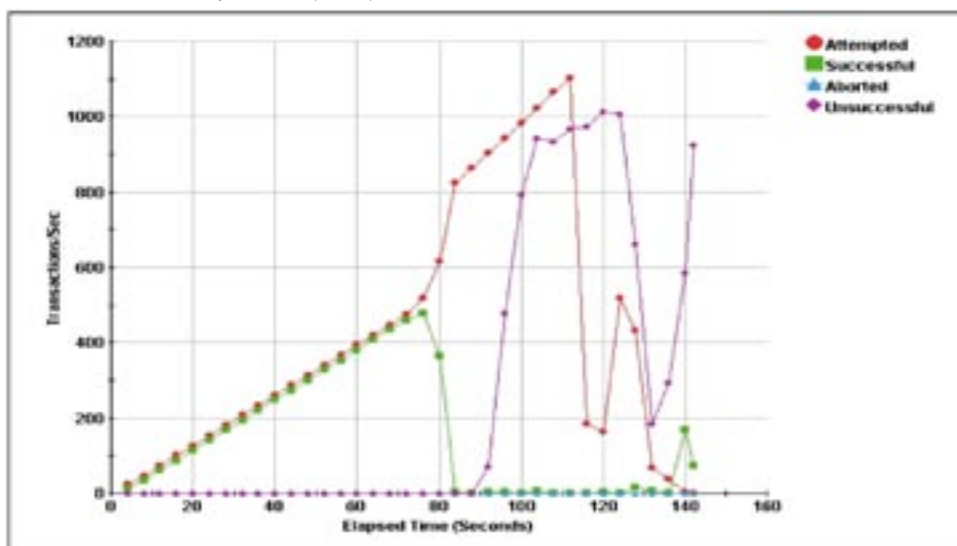


Figure 5: Effect on CPS with a multi-protocol test (HTTP and FTP).

7. Measure drop in response times due to FTP loading.

As the firewall becomes stressed with increasing CPS, *processing and response latencies through the firewall increase*. Since the firewall is usually a gatekeeper to the web-servers, increased latencies equate to slower user-experience that in some cases grows to unacceptable access times for your websites (Figure 6).

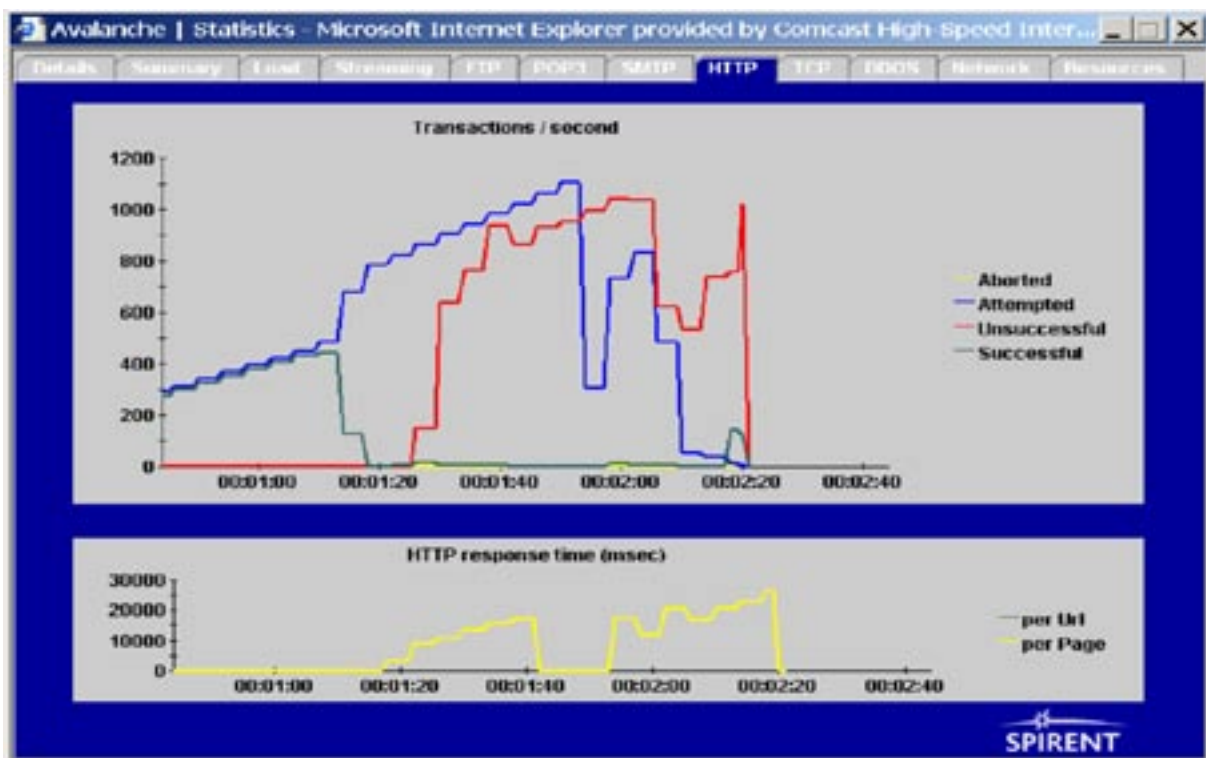


Figure 6: Drop in response times due to FTP added to HTTP transactions.

8. Test for maximum concurrent open connections.

In addition to CPS testing (steps 5 and 6), Concurrent open connections is a key firewall benchmark parameter. To create this test:

- Take the "Firewall Stress" test and copy it to "FW Open."
- Change the load profile to reflect **SimUsers** as the load rather than connections per second. Put in 40 SimUsers as the starting number of concurrent users. Increase the number of SimUsers gradually till you see failing tests.

In the sample-run, a moderate increase in SimUsers (over 45) resulted in a large number of transactions timing-out. Incomplete transactions reduce availability of the servers and can degrade performance to unacceptable levels.

This concludes the “Firewall Stress” test. As discussed earlier, you are encouraged to go beyond CPS and open connections to test relevant criteria such as packets per second and throughput using multi-protocol tests.

4.4 “Firewall Load” – Load Testing Your Firewall

4.4.1 Objective

Determine firewall operating limits during multiple IP protocol-based traffic and Distributed Denial of Service (DDoS) attacks: Does your firewall remain available while under load and under attack?

4.4.2 Additional Requirement

All-protocol software bundle including Inline DDoS package

4.4.3 Running “Firewall Load”

1. Select and copy “Firewall Stress” to “Firewall Load.”

- Create a traffic load that represents your network traffic with multiple protocols.

For example, you could create multiple weighted user-profiles to create a traffic mix comprising HTTP (20%), FTP (20%), SMTP (40%), RTSP (20%) and HTTPS (20%). The assigned weights are arbitrary – select them to reflect the realistic traffic mix through your firewall.

- **Reflector**-side configuration needed
- Re-run the modified “Firewall Load”:
 - Modify the load spec, if needed.
 - Ensure that there are no failing tests.

2. Re-configure your firewall to enable multiple protocols.

The following is an *example* of the type of firewall configuration changes needed for this test:

```
static (inside,outside) 10.10.10.10 10.10.10.10
static (inside,outside) 10.10.10.11 10.10.10.11
static (inside,outside) 10.10.10.12 10.10.10.12
static (inside,outside) 10.10.10.13 10.10.10.13
mailhost (inside,outside) 10.10.10.14 10.10.10.14 10 11
conduit (inside,outside) 10.10.10.10 80 tcp 0.0.0.0 0.0.0.0
conduit (inside,outside) 10.10.10.11 21 tcp 0.0.0.0 0.0.0.0
conduit (inside,outside) 10.10.10.12 554 tcp 0.0.0.0 0.0.0.0
conduit (inside,outside) 10.10.10.13 443 tcp 0.0.0.0 0.0.0.0
```

```
conduit (inside,outside) 10.10.10.14 25 tcp 0.0.0.0 0.0.0.0
route outside 192.168.0.0 255.255.255.128 192.168.0.2 1
route outside 192.168.0.128 255.255.255.128 192.168.0.129 1
timeout xlate 24:00:00 conn 12:00:00 udp 0:02:00
timeout rpc 0:10:00 h323 0:05:00 uauth 0:05:00
```

3. Run “Firewall Load” and monitor real-time results on the **Avalanche**.

For example, click on HTTP (Figure 7) and RTSP (Figure 8). **Avalanche** displays real-time statistics on a per-protocol basis at run-time as well as creates a spreadsheet with the final results after the test-run.

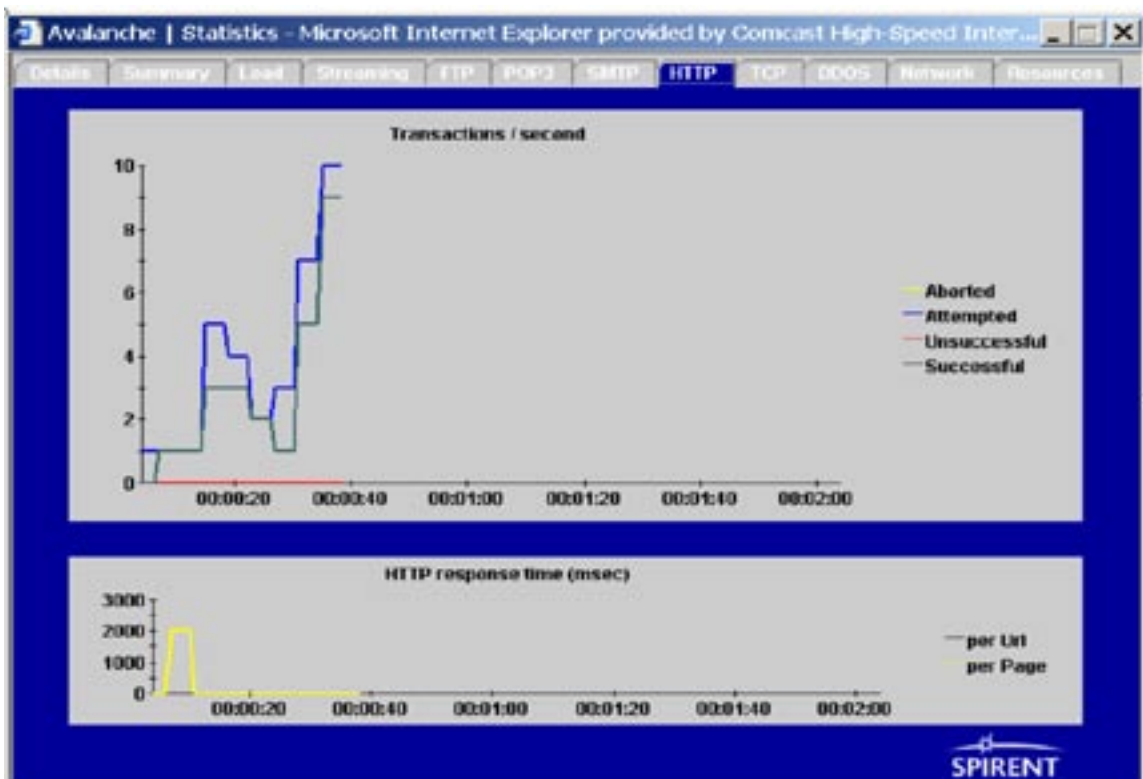


Figure 7: “Firewall Load” real-time results for HTTP.

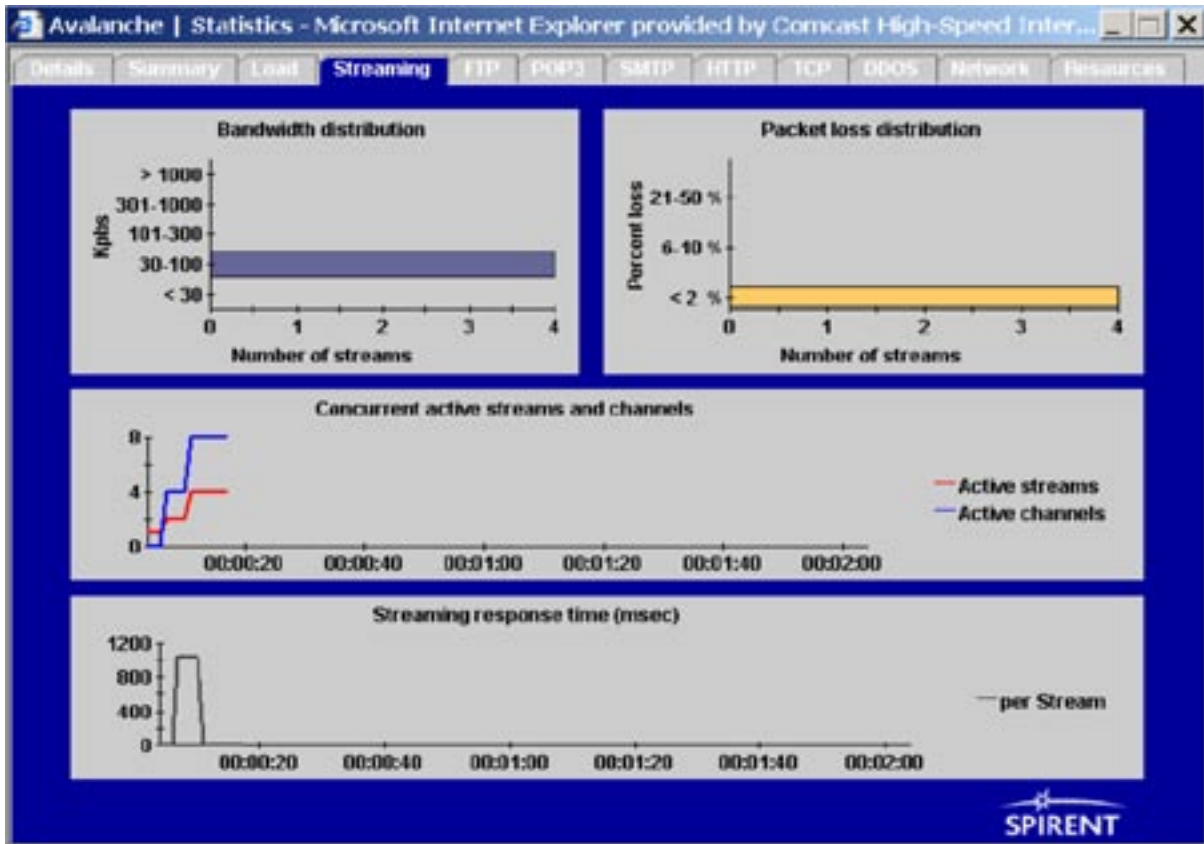


Figure 8: “Firewall Load” real-time results for RTSP.

4. Increase number of users to determine performance degradation.

Using page response times for the protected web-servers (simulated on the *Reflector*) or stream response times simulated on media servers, increasing the number of virtual users (*Simusers*) on the *Avalanche* can quickly determine the threshold points.

It is not adequate to merely increase the number of virtual users alone. “Realism testing” demands that the large number of users also use different IP-protocols that reflect their unique application accesses. *Avalanche* is best suited to determine if the latency per protocol per user through your firewall meets your response-time requirements. It is entirely likely, that under “realism testing” using *Avalanche*, you will uncover conditions wherein the traversal latency through a firewall serving 40 concurrent users on diverse protocols may be acceptable. However, as shown in Figure 9 below, 45 or more concurrent users could easily translate to very high and erratic response times for even a most common protocol mix consisting of HTTP, HTTPS and FTP based applications!

If the same conditions were to be extended to include RTP/RTSP based multi-media server access, Telnet, DNS, SMTP and other IP-protocols (all supported by the *Avalanche*), the processing latency through the firewall – and hence the overall access latency for the end-applications on these protocols – could be significantly worse.

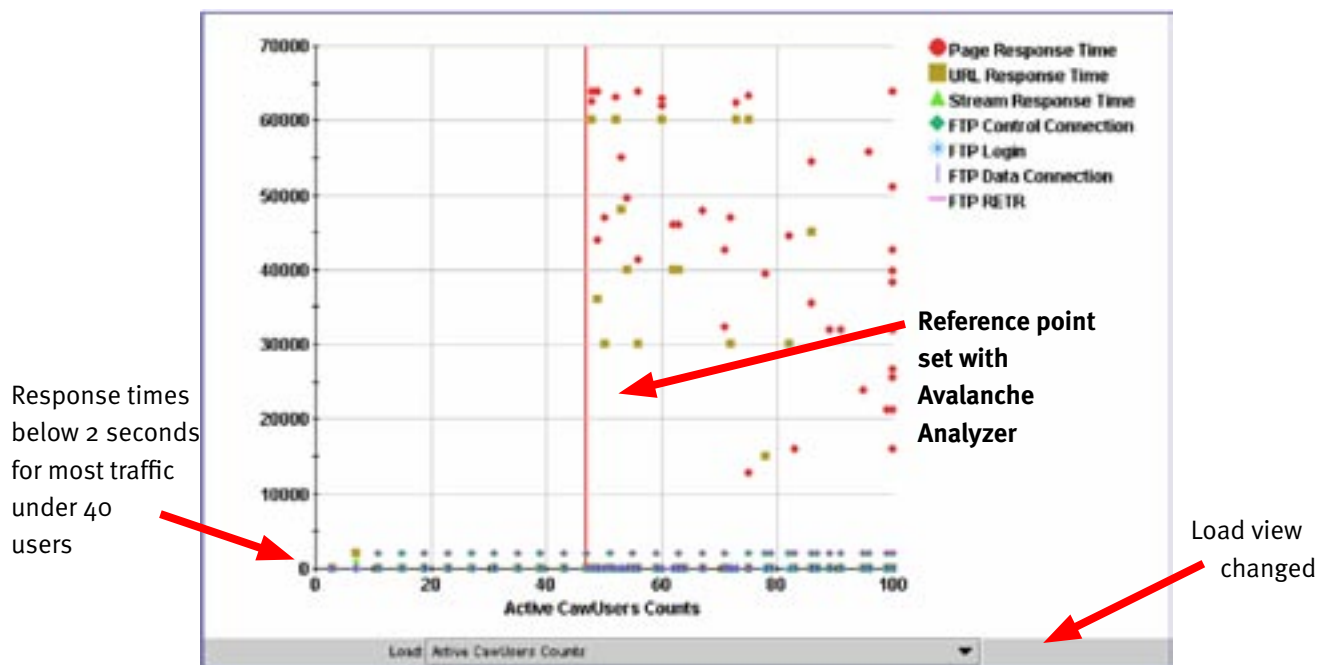


Figure 9: “Firewall Load” performance degradation with increasing users.

Readers are encouraged to extend their “Firewall Load” tests to cover more users and additional IP-protocols to determine if their Firewall can indeed meet their Firewall transaction (and hence overall application access) latencies.

5. Add simulation of DDoS attacks.

Most firewalls today are subject to well-known attacks from hackers attempting to bring down your network. DDoS attacks using spoofed IP addresses are the most damaging, and continue to appear in new forms. **Avalanche** enables incorporating DDoS attacks as part of the traffic mix incident on the firewalls – further increasing the level of network realism.

You can now test for the effect of bad DDoS traffic affecting “good throughput” by varying the traffic mix as follows:

- Keep DDoS traffic to a known constant (e.g., 5%) while varying the multi-protocol load of good throughput (e.g., from SMTP:FTP:HTTP:HTTPS in 45:15:30:10 ratios to a different composition with the same protocols)
- Vary DDoS traffic (e.g., from 3, 5, to 8%) of total traffic while keeping the percentage-mix of good throughput the same (constant component ratios)
- Vary both – alter the levels of DDoS traffic while altering the relative percentage of protocols constituting good throughput.

These three independent tests could yield very insightful findings on whether your firewall continues to be available while under attack, and if the transit-latency through the firewall is maintained at acceptable levels.

The following flow illustrates one example of running the tests above.

- To include burying DDoS attacks as part of realistic traffic, simply enable the “Inline DDoS” option in the “Firewall Load” test on your **Avalanche** (Figure 10).
- You will see a list of well-known tests such as *Ping of Death*, *Smurf*, *SYN floods* and others.
- Click on each of these as needed and program variables pertinent to each attack. Since each type of attack can be programmed to begin at arbitrary instances in the test, and variables per attack vary, see the **Avalanche** user manual for details on programming each attack.

In addition, you can use a scripting feature to manipulate bit-level detail in each packet in the Ethernet frame, thereby creating your customized attack, if needed.



Figure 10: Example of configuring a DDoS attack using the Inline DDoS option.

Using the same firewall (100 Mbps throughput) as in prior tests, Figure 11 shows how even modest amounts (in percentage of good throughput) of DDoS traffic can cripple your firewall availability. Note that during a 4-second interval (in this case, from 02:50 to approximately 02:56), no new TCP connections could be established while under attack!

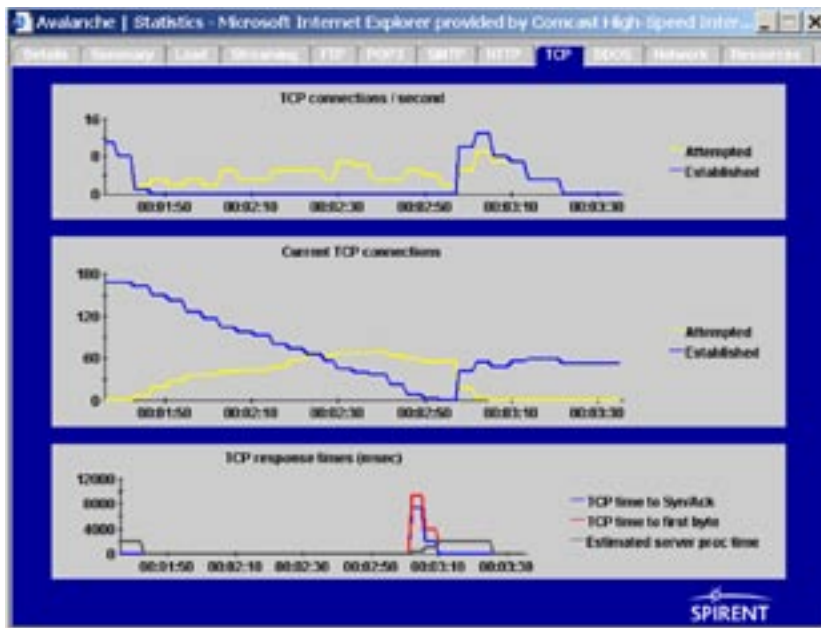


Figure 11: Degradation of TCP performance while under DDoS attack.

If the firewall is unable to establish connections for a long period of time, it is essentially rendered unavailable during that period – even if the firewall were to recover after the attack (Figure 12). This is an important finding that will help you measure your firewall availability while under attack, and is even more critical if your firewall becomes a choke-point for your overall applications and infrastructure availability.

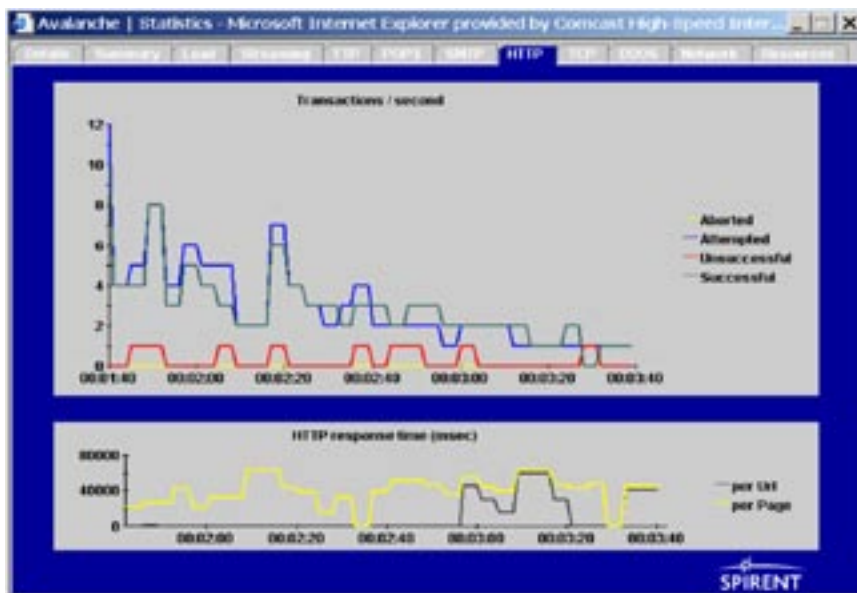


Figure 12: Degraded response times while under DDoS attack.

The following findings are noted from running “Firewall Load” including DDoS attacks:

- HTTP and FTP stopped completely during DDoS!
- Some RTSP streams remained active and TCP started to send Resets.
- The particular firewall seems to have a prioritized handling of SMTP/e-mail while under attack. Therefore, though HTTP and FTP processing was essentially halted, the firewall continued to process mail.
- Some firewalls may shut down completely instead of recovering. Running DDoS-injected tests for longer durations (48-96 hours) is recommended since perimeter firewalls today are subjected to routine attacks from the public internet.
- Even after the attack, the recovery profile of a firewall may leave more to be desired. For example, as shown in Figure 12, even with recovery, the HTTP response times stretched out to greater than a minute – which is unacceptable for e-commerce types of applications.

This completes the discussion on creating basic, stress and load tests for firewall evaluation. The key findings using this test methodology with a 100 Mbps firewall and **Avalanche/Reflector** technology are:

- CPS for HTTP is greater than 2,400 but the addition of one more protocol (FTP) cuts this down by 75%.
- Heavy CPS results in traffic slow-down, which may be unacceptable.
- Open Connections at 120,000 – *however* many transactions were incomplete.
- Mixed traffic will meet Service Level Agreements (SLA) at 40 users. Supporting additional users requires an upgrade to a higher performing firewall.
- Firewall recovers from DDoS but *does not allow* HTTP or FTP traffic during attack. Mail traffic is kept safe, since this particular firewall assures SMTP throughput as a priority.

5. Summary

This application note has shown how to go beyond the typical method of extrapolating measurements to evaluate firewalls by conducting a series of tests to accurately measure firewall availability while under load. The most important findings from these tests ensure that your firewall provides both security and availability:

- Validating performance as applicable to your network
- Data throughput (proven Mbps/Gbps)
- Data forwarding rate (proven PPS)
- Concurrent TCP connection capacity (maximum entries in the connection table)
- Availability while under DDoS attacks

Most important to these findings is recognizing that extrapolation alone is not enough to adequately evaluate firewall performance. Relying solely on extrapolation to prove that a firewall meets corporate security needs is a disaster waiting to happen when it becomes obvious that the firewall breaks down. The following table summarizes the risks of using lab testing and extrapolation versus real-world behavior.

Lab	Real-world	Risks
Single device	Multiple devices	Interoperability End-to-end security and availability
Controlled traffic	Complex traffic	Uncharacterized performance, security and availability under multiple-protocols
Few security policies	Layered policies	Behavior under multiple policies
Simple network	Complex Internet behavior	Latency, packet loss (others)
Controlled load tests	Variable loads	Performance under widely-varying loads

Table 1: Risks associated with lab testing vs. real-world conditions.

Following a firewall testing methodology consisting of the *basic, stress and load tests* as discussed in this paper enables you to identify and mitigate risks associated with your firewall evaluations. Combining the knowledge of your network and security engineers with the capabilities of the Spirent **Avalanche**, **Reflector**, and **SmartBits** products enables you to evaluate your firewall with the *realism of production networks with the precision of lab testing*. Doing so ensures you have the right firewall product to support your corporate network security needs and helps increase your infrastructure security and availability.

Readers are encouraged to extend the basic, load and stress-test oriented methodology discussed in this paper to include additional feature-testing for vendor-specific firewall features such as deep-packet inspection, application-layer firewalls and aggregated firewall platforms that include IDS, IPSec-VPN and other security functions.

Contact your nearest Spirent Systems Engineer to help you extend the methodology in this paper to help you include your specific “realism testing” parameters to get the most benefit out of your Spirent Firewall and end-to-end testing methodologies.

6. Background

There are numerous excellent sources for Firewalls, Firewall-testing methodologies and new trends in the Firewall market available at vendor websites, security user-groups and third-party certification labs.

Some background information that may be useful to the reader:

How Firewalls Work, Tyson, J., www.howstuffworks.com

Internet Firewalls: Frequently Asked Questions, Curtin, M. and Ranum, M.J., December 1, 2000, Revision 10.0, www.interhack.net/pubs/fwfaq

Benchmarking Methodology for Firewall Performance, IETF RFC 3511, <http://www.ietf.org/rfc/rfc3511.txt?number=3511>

Firewall Performance Testing Methodology, Joung, P and Pochiraju, A, Spirent Communications, <http://www.spirentcom.com/documents/I062.pdf>