

**The Fundamentals  
of  
Intrusion Prevention System Testing**



New network-based Intrusion Prevention Systems (IPS) complement traditional security products to provide enterprises with unparalleled protection against external and internal attacks. An exponential rise in application vulnerabilities that are easily exploited through standard ports have rendered traditional Firewalls ineffective against attacks. While Intrusion Detection Systems (IDS) can often detect these attacks, these passive systems offer little more than an after-the-fact notification. In contrast, an IPS is designed to examine all traffic that passes through it to detect and filter out malicious packets. Analogous to anti-virus systems, IPS's can be centrally managed and armed with additional filters whenever a new vulnerability is discovered.

More akin to switches than sensors, IPS's are typically installed as part of the network infrastructure, both at the perimeter and in the core of the network. A typical IPS deployment is shown in figure 1. In this example, three IPS's filter packets on one external and two internal segments. The internal segments in this example are different subnets, connected by a router.

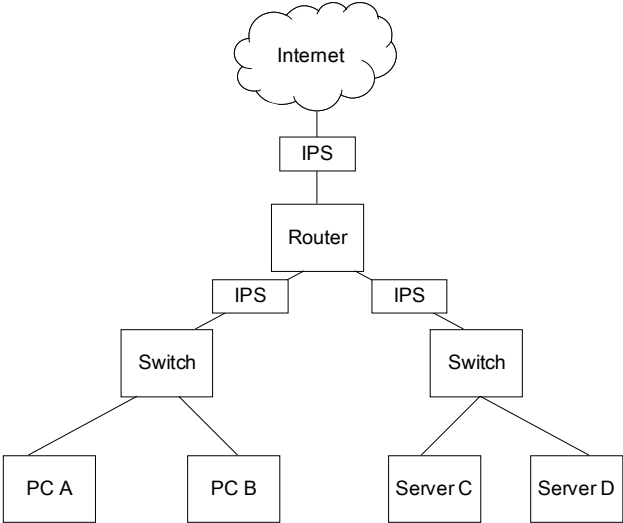


Figure 1: Prototypical IPS deployment.

To be effective, an IPS must exhibit the same network performance characteristics of other network infrastructure products, like switches and routers, while at the same time performing stateful deep packet security processing to filter out layer 2-7 attacks. This stringent set of requirements demands a new test methodology and a new set of tools to verify their efficacy. In particular, it is necessary to measure performance and security functions simultaneously since they are interdependent. In this document we define a methodology that should be used when testing IPS products.

## IPS Testing Principals

### Traffic Mix

As with any network infrastructure product, it is necessary to test the IPS with real-world traffic mixes. This is important because packet size, protocol distribution, packet contents, packets per second, protocol mix, the number of sessions, new sessions per second, and duration of sessions can all impact system performance. In particular, IP networks typically carry a combination of TCP and UDP packets with a small fraction of other traffic types. TCP flows are capable of adapting to network congestion, while UDP tends to be less resilient. Testing with only one traffic type is not representative of real-world conditions and tests results will vary dramatically between the lab and a production environment.

Studies by CAIDA, the Cooperative Association for Internet Data Analysis, have characterized traffic on the Internet. These studies show an average packet size between 413 and 474 bytes.<sup>1,2</sup> The distribution of packet sizes is tri-modal, with peaks near 64, 512, and 1518 bytes. About 85% of the packets and 92% of the bytes are TCP, and 12% of the packets and 5% of the bytes are UDP. The remaining traffic is a mix of IP encapsulated, ICMP, GRE, and other protocols. Two-thirds of the UDP traffic in these studies was DNS and RealAudio. The remaining is a mix of applications, 13% of which CAIDA could not classify. The average packet size of a UDP packet was 182.3 bytes. The traffic in Local Area Networks varies significantly from that of the Internet, which CAIDA monitored. In particular, the amount of UDP traffic in local area networks can be significantly higher than found on the Internet. Many RPC applications use UDP instead of TCP for performance reasons. Microsoft name resolution services, Sun RPC applications, SNMP, DNS, and streaming media applications can all increase the UDP workload in a LAN environment. For example, if the corporate network is using Sun NFS, UDP can account for a significantly higher percentage of the workload and an increased average UDP packet size.

As stated above, an IPS is deployed both at the perimeter and in the corporate backbone. When testing an IPS, it is important to test the device's performance under a mix of TCP and UDP traffic. Testing a range of workloads with a TCP:UDP traffic mix of 100:0 (all TCP), 90:10, 80:20, and 70:30 is recommended. These ratios are by bytes; 90:10 means that 90% of the bytes are TCP and 10% of the bytes are UDP. The average UDP packet size should be around 200 bytes. An IPS that can perform well in these ranges is likely to perform well in any corporate network environment.

### Throughput and Latency

In testing an IPS, it is impossible to separate throughput from latency. As an infrastructure element, latency will dramatically impact throughput and can quickly

---

<sup>1</sup> [http://www.caida.org/analysis/AIX/plen\\_hist/](http://www.caida.org/analysis/AIX/plen_hist/)

<sup>2</sup> <http://www.caida.org/outreach/resources/learn/trafficworkload/>

become the dominant parameter for network performance. This effect is particularly noticeable for applications built on TCP.

TCP achieves a maximum throughput of  $window/RTT$ , where *window* is the maximum window size and *RTT* is the round trip latency in the network. In the TCP protocol, 16 bits are reserved for the window size in the TCP header, which allows for *window* sizes up to 64 kilobytes. In a local area network, *RTT* is typically around 1 millisecond (msec). This gives a maximum throughput of 64 Mbytes/sec, or 512 Mbps.<sup>3</sup>

Any network element that adds a significant amount of latency will reduce TCP throughput. A network element will increase *RTT* by twice the one-way latency, which is typically what is reported on manufacturer's datasheets. For example, adding a network element with a latency of 2 milliseconds will increase *RTT* from 1 msec to 5 msec, reducing maximum throughput by 80% – about 100 Mbps.

The latency of an IPS can be multiplied several-fold in a real deployment. As shown in figure 1, a packet may traverse multiple IPS's between PC A and Server C. Each traversal of an IPS introduces latency and the effect is cumulative. For example, if each IPS introduces 2 milliseconds of one-way latency, *RTT* jumps to 9 msec and the maximum bandwidth reduced to less than 57 Mbps.

Latency must be measured along with bandwidth under steady-state conditions. To illustrate the pitfalls of separately testing latency and bandwidth, one customer tested an IPS product by sending a 10 second burst of traffic at 1 Gbps using SmartBits. They observed no packet loss, and concluded that the product had a throughput of 1 Gbps. Closer examination revealed that the latency climbed to almost 30 *seconds* during the test; the 10-second burst of traffic was absorbed by a buffer in the IPS and drained out of the product over the next 30 seconds.

In summary, latency and throughput must be measured simultaneously, and the average latency should be similar to that of other infrastructure elements – well under a millisecond.

## IPS Configuration

For any performance tests to be meaningful, it is important that the IPS be configured with all filters enabled. New vulnerabilities are being discovered at an exponentially increasing rate; the number of reported vulnerabilities has doubled every year since 1998<sup>4</sup>. This means that the number of filters required to protect a network will increase dramatically over the lifetime of the IPS.

---

<sup>3</sup> To work around this limitation a special option, called the TCP window scale option, was introduced in RFC 1323. This option is negotiated at the opening of the connection, so if a window size of greater than 64 KB is to be established it must be done at connection set-up time. Unfortunately, many applications do not set this option, so their throughput on a LAN is governed almost entirely by the latency in the network.

<sup>4</sup> <http://www.cert.org/stats/>

In some IPS architectures, performance is directly proportional to the number of active filters. Software-based IPS solutions are particularly susceptible to this phenomenon. Each filter introduces additional rules that must be processed for each packet that flows through the box. Hardware-based solutions may incorporate specialized accelerators and parallel processing engines that allow the number of filters to scale without impacting performance.

To ensure that the IPS can scale to the number of filters that will be needed after several years of service, performance testing should be performed with all filters enabled.

## Attack Blocking

Unlike an IDS, IPS's are active, inline devices. This gives the IPS the opportunity to block attacks, but introduces new requirements on testing.

Intrusion Detection Systems (IDS) are traditionally tested by replaying packet traces of captured attacks and verifying that the IDS generates an appropriate alert. Since IDS's are passive devices, they inspect a copy of network traffic; the copy is typically obtained using port mirroring to inspect traffic going through a switch. IDS test tools mimic this arrangement by replaying the attacks *unidirectionally* at the IDS.

When testing an IPS, attacks must be played bi-directionally. That is, packets from the attacker should arrive one IPS interface, and packets from the target on another. Most IDS testing tools play attacks unidirectionally, with two notable exceptions. Blade Software is purported to be releasing a tool that can replay attacks bi-directionally<sup>5</sup>, and the open source tool "tcpreplay" can play attacks bi-directionally.

No matter what tool is used, the tool must independently verify that the attack is blocked and retransmit lost packets. The only reliable way to verify that a replayed attack is blocked is to ensure that the attack packets are not received at the target. We know of no IDS/IPS test tool that checks whether the replayed attack is actually blocked.

The test tool must include a retransmission mechanism to allow for packets lost due to network congestion. Since the testing should be done under load, it is possible that the IPS drops a packet due to congestion. The tool that plays the attacks must retransmit lost packets to account for this possibility.

The attacks chosen for IPS testing must be attacks that can be blocked. Detecting an attack without blocking is an IDS function, and should be reserved for IDS testing.

Finally, the test must verify that legitimate traffic is not blocked. The best way to do this is to play a high load of clean background traffic while performing security testing and verifying that it is minimally affected by the attacks and never inadvertently blocked. Like IDS's, some IPS architectures may miss attacks if the attack is launched while the

---

<sup>5</sup> The bidirectional version is in beta.

system is heavily loaded. Load can take many forms, including bandwidth, packet arrival rate, session count, and session creation rate. For this reason, the device's ability to block attacks should be tested at and near the physical limits of the IPS, not just the rated limits.

In addition to testing security efficacy, testing at the rated limit has an important side effect: it tests whether the IPS will erroneously block good traffic while under attack. Although some performance degradation can be expected if the IPS is under an extremely heavy attack load, the degradation should be graceful. IPS performance should not "fall off a cliff" because the load is high, and it should never crash. Any such instability is inviting attackers to launch denial of service attacks against the network the IPS is supposed to protect.

To summarize, IPS security testing should be done with a high load of attack-free background traffic. The testing should show that this background traffic is minimally affected by the tests and never inadvertently blocked. The tool used for replaying attacks must detect and retransmit lost packets, and it must report when an attack completes or is blocked, independently verifying the IPS's claim that it blocked the attack. To our knowledge, no previously available testing tool meets these requirements. Spitfire, a tool described below, was developed to fill this gap.

## Proposed Test Setup and Methodology

A specific test setup and methodology is proposed based on the test principals described above. The test setup is straightforward and representative of a real-world environment. It supports various tuning knobs so that the test scenarios can be tailored to match specific customer environments.

The test jig is illustrated in Figure 2. A pair of switches that are used to aggregate multiple data sources flanks the IPS. Depending on the IPS capacity, one or more switches may be used to achieve the desired bandwidth.

In our lab, we use four PCs and a SmartBits to generate background and attack traffic, and to measure throughput and latency.

The PCs are 2.8 GHz Pentium 4s running RedHat 7.3 (Linux 2.4.18-3). Each machine has three network interfaces: two Intel PRO-LAN 10/100/1000 Ethernet adapters for data traffic and one management Ethernet interface. We cabled the data port of the machines and assigned IP addresses as shown in figure 2. All links are Gigabit Ethernet.

To generate TCP traffic load, we ran Apache 1.3.23 on the machines 1-3. We installed a large (50Mbyte) text file containing random data in the ServerRoot directory. Each machine retrieved this file via HTTP from a corresponding server on the other side of the IPS. For example, machine 1 retrieved a file via the 10.0.1.1 port from 10.0.2.2.

Machine 2 similarly retrieved a file from machine 3, and machine 3 from machine 1<sup>6</sup>. We wrote a simple program called *netgen* to fetch multiple files from the web server in parallel and to read the generated data at a fixed rate. Command line parameters control the rate at which data is consumed and the number of simultaneous streams. This simple technique can easily generate more than 2 Gbps of TCP traffic with a few PCs.

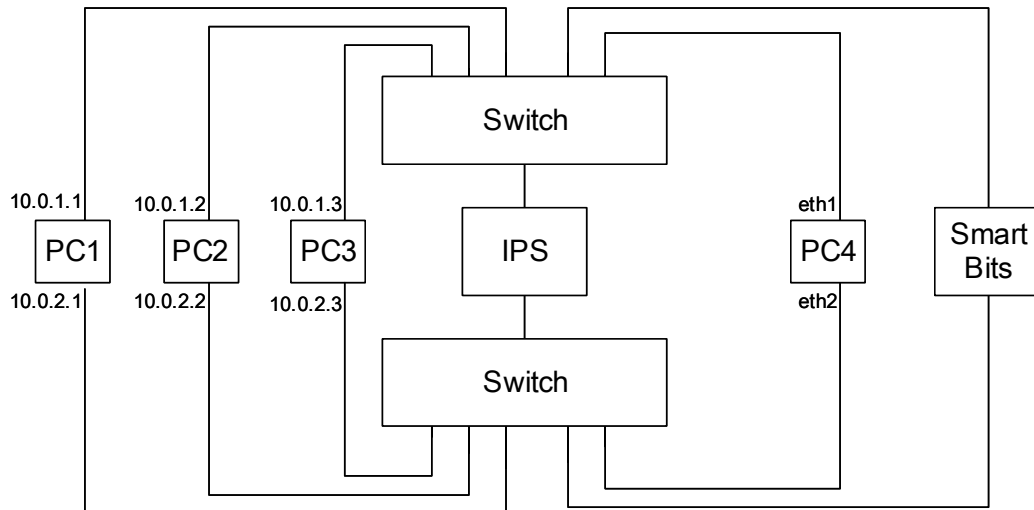


Figure 2: Test Jig Configuration

To measure the amount of TCP traffic actually generated, we wrote a program to sample the counters on the Ethernet adapters every two seconds and total the results. The program computes mean bandwidth over 10 and 60 seconds intervals.

To generate UDP traffic, we used a SmartBits 6000B with a LAN-3301A 10/100/1000 Mbps TeraMetrics module. The 6000B was controlled using SmartFlows 2.0 software, and both average latency and UDP throughput were measured. We configured the SmartBits to send 192 byte UDP packets with random payload on ports 1024 and 1025 through the IPS.

To calibrate the system for a given traffic mix, we set the TCP and UDP flows for the desired traffic mix. For example, to generate 1000 Mbps with an 80:20 mix, we want to generate 800 Mbps of TCP and 200 Mbps of UDP. SmartBits is therefore configured to generate 200 Mbps of UDP and netgen is configured to generate 800 Mbps of TCP.

To verify the configuration, we bypass the IPS with a wire and measure the aggregate throughput (TCP and UDP) as well as latency. The baseline latency is that introduced by the switches and is typically small (in the tens of microseconds). We then reintroduce

---

<sup>6</sup> To force the traffic out the right interface, we installed specific routes on each of the PCs.

the IPS and measure latency and throughput. Three samples are taken to ensure repeatability. All tests are conducted with all attack filters enabled.<sup>7</sup>

## Security Testing

The most important feature of an IPS is that it blocks attacks while not adversely affecting network performance. This means it must pass the performance test above, and that it must not miss attacks when under load.

To test this property, we ran security tests under each load scenario listed above. The Spitfire tools (described below) are used to replay attacks through the IPS on PC 4. These tools are described below. The attacks are stored in packet traces, and Spitfire replays the attack exactly as they would appear on a real network. Spitfire is instrumented to log when an attack is missed by the IPS (i.e., when the network packets containing the attack are seen on the victim). The number of missed attacks is determined by counting the number of completed pcaps (packet captures). The script included in Appendix A can be used to drive Spitfire and count the number of blocked attacks.

## Spitfire

Briefly, Spitfire divides a packet trace into two parts: those generated by the *attacker* and those generated by the *victim*. Spitfire parses the packet trace (called the *pcap*) one packet at a time. The first time an IP address is seen in a file, the IP address is "assigned" to the attacker if it is in the IP source address field of the packet, or assigned to the victim if it is in the destination field. For instance, consider a pcap consisting of a standard three-way TCP handshake contains 3 packets:

Packet 1 (SYN):	ip.src = 172.16.5.5	ip.dest = 172.16.5.4
Packet 2 (SYN-ACK):	ip.src = 172.16.5.4	ip.dest = 172.16.5.5
Packet 3 (ACK):	ip.src = 172.16.5.5	ip.dest = 172.16.5.4

When spitfire reads the first packet, the address 172.16.5.5 is encountered for the first time in the source field, and the address 172.16.5.4 is encountered for the first time in the destination field. The address 172.16.5.5 is therefore associated with the attacker, while the address 172.16.5.4 is associated with the victim.

When it comes time to replay the attack, victim packets are transmitted on eth2, and attacker packets are transmitted on eth1. To replay the sequence above, Spitfire begins by sending packet 1 (an attacker packet) over eth1. When this packet arrives on eth2, it sends packet 2 out eth2 and waits for packet 3 to arrive on eth1. When the packet arrives, Spitfire sends packet 3 on eth1. When the last packet arrives on eth2, Spitfire outputs that it has completed the pcap.

---

<sup>7</sup> In some IPS products, we found that certain filters would block legitimate traffic, such as the HTTP transfers. These filters were disabled individually and the effect was noted.

If a packet is lost, the sender retries after a timeout period (typically every 2 seconds). The sender infers that the packet is lost if it does not receive the next packet in sequence within the timeout. For example, if Spitfire sends packet 2 on eth2 and does not receive it on eth1 within the timeout, it resends packet 2. If progress is not after a specified number of retransmissions, the session is aborted and Spitfire outputs a message indicating that the session has timed out.

To ensure that the packet is correctly routed through the switches, the Ethernet MAC addresses are rewritten when the packet is sent. In addition, the IP addresses are also rewritten and the packet's checksums updated accordingly. Thus, in the example above, when Spitfire sends packet 1, the IP source address of the packet that appears on the wire is 10.0.0.1, and the IP destination address is 10.0.0.2<sup>8</sup>. Spitfire writes the modified packet directly to the Ethernet driver using a raw socket.

Within the context of an IPS, if Spitfire reports that the pcap containing the attack has timed out, the IPS has correctly blocked the attack. If Spitfire reports that the pcap has completed, the IPS missed the attack, regardless of what the log indicates.

## Conclusion

This paper has presented several characteristics that must be evaluated when testing an IPS. Throughput, latency, and attack blocking must be simultaneously measured. Throughput and latency should be on par with other pieces network equipment; in a LAN, this means Gbps throughput with an average latency of a few hundred microseconds. The test tool must independently verify that the tested attacks are blocked (or not). Only attacks that can be blocked should be tested – attacks that can only be detected, but not blocked, belong in the domain of IDS testing. To assure scalability, the IPS should be tested with all filters enabled.

We have presented a test jig and tool, called Spitfire, that shows how this testing can be accomplished with a modest amount of equipment. Contact TippingPoint Technologies for more information about Spitfire.

---

<sup>8</sup> The mapping of IP addresses can be controlled by a command line parameter to Spitfire.

## Appendix A

The following script is used to replay attacks in the Spitfire toolset. The only parameter to the script is the unique identifier for the test (e.g., "TCP80-first"). Six sample packet trace files are shown – you can use as many as you have (we used 773 in the Tolly test).

```
#!/bin/bash
run=$1
spitfire -t 1000 -R 5 \
    -f pcaps/DV536/pcap.0027.1 \
    -f pcaps/DV536/pcap.0032.1 \
    -f pcaps/DV536/pcap.0034.1 \
    -f pcaps/DV536/pcap.0035.1 \
    -f pcaps/DV536/pcap.0036.1 \
    -f pcaps/DV536/pcap.1482.2 \
> $run.log
```

This script records attack completion and timeout information to the file \$run.log. To determine the number of attacks blocked, the following command is used:

```
grep timeout $run.log | wc -l
```

The number shown should be the same as the number of pcaps in the attacker script.