Juniper Networks IDP 600F V3.1

Technical Evaluation

An NSS Group Report

First published July 2005 (Version 1.0)

Published by The NSS Group
Security Testing Laboratories
Mas la Carrière, Route de Ganges
30440 Sumène, France

Tel : +33 (0)4 67 81 49 11
E-mail : info@nss.co.uk
Internet : http://www.nss.co.uk

TABLE OF CONTENTS

TABLE OF FIGURES

The NSS Group

The NSS Group is the world's foremost independent security testing facility.

With British headquarters, and security and network infrastructure testing facilities in the South of France, The NSS Group offers a range of specialist IT, networking and security-related services to vendors and end-user organisations world-wide.

The NSS Group's Security Testing Laboratories are available to vendors and end-users for fully independent testing of networking, communications and security hardware and software.

The NSS Group also operates certification schemes for vendors and certification bodies, and currently provides evaluation and certification of a wide range of security products, including IDS/IPS appliances, firewalls, VPNs, Web Application firewalls, multi-function security appliances, cryptographic devices and PKI products.

Output from the labs, including detailed research reports, articles and white papers on the latest network and security technologies, are made available on the NSS web site at http://www.nss.co.uk.

The NSS Group awards are recognised world-wide as being the most desirable and essential when it comes to security products. Vendors consider the awards to be a crucial step in any security-related marketing campaign, whilst feedback from readers of the reports indicates that participation in an NSS Group test and/or one of the **NSS Approved** awards is a prerequisite for any security product in order to be considered for purchase.

Foreword

Following the huge success of the first comprehensive *Intrusion Prevention System* (IPS) test of its kind, The NSS Group is pleased to present the results of its third IPS Group Test, the largest so far, which includes a number of new products not included in the first two reports.

As with the first two Editions, this exhaustive review will give readers a complete perspective of the capabilities, maturity and suitability for immediate deployment of each of the products tested. The NSS Group established this test as IPS products are being actively deployed as a new layer in defence-in-depth security architectures.

The NSS IPS Group Test evaluates the performance, reliability, security effectiveness, and usability of Network IPS products. The test consists of seven sections within three primary areas: *performance and reliability, security accuracy,* and *usability*.

Overall, the brand new test suite contains over **800 individual tests**, many of which are run multiple times, to provide the most thorough and complete evaluation of IPS products available anywhere today. The NSS Group has developed advanced testing methodologies for both *Rate-Based IPS* and *Content-Based IPS* products, since these devices are often very different in operation, although all products tested in this edition of the report are content-based.

It is worth pointing out that not every product submitted for testing receives an *NSS Approved* award. Standards are very high, and only those appearing in this report have received ***NSS Approved*** awards. For this latest edition, **ten** vendors submitted a total of **twelve** products for testing, and **eight** of these passed our stringent testing to receive **NSS Approved**. It is heartening to note that this is a much-improved success ratio over Edition 2.

We believe that our IPS test methodologies - which have been updated again for this test - will become the *de facto* standard for testing in-line Intrusion Prevention/Attack Mitigation devices, and the *NSS Approved* logo an essential item on the list of requirements when purchasing these products.

We also believe that this report is essential reading for anyone considering deploying Intrusion Prevention Systems in their networks, either in a test or live situation, and we hope that you find it both informative and useful in making your purchasing decisions. The latest **IPS Group Test** report can be viewed on-line at *www.nss.co.uk/ips*

Bob Walder

INTRODUCTION

In a survey commissioned by VanDyke Software, some 66 per cent of the companies who responded said that they perceive system penetration to be the largest threat to their enterprises.

The survey revealed that the top eight threats experienced by those surveyed were *viruses* (78 per cent of respondents), *system penetration* (50 per cent), *DoS* (40 per cent), *insider abuse* (29 per cent), *spoofing* (28 per cent), *data/network sabotage* (20 per cent), and *unauthorised insider access* (16 per cent).

Although 86 per cent of respondents use firewalls (a disturbingly **low** figure in this day and age, to be honest!), it is apparent that firewalls are not always effective against many intrusion attempts. The average firewall is designed to deny clearly suspicious traffic - such as an attempt to telnet to a device when corporate security policy forbids telnet access completely - but is also designed to allow some traffic through - Web traffic to an internal Web server, for example.

The problem is, that many exploits attempt to take advantage of weaknesses in the very protocols that **are** allowed through our perimeter firewalls, and once the Web server has been compromised, this can often be used as a springboard to launch additional attacks on other internal servers. Once a “rootkit” or “back door” has been installed on a server, the hacker has ensured that he will have unfettered access to that machine at any point in the future.

Firewalls are also typically employed only at the network perimeter. However, many attacks, intentional or otherwise, are launched from within an organisation. Virtual private networks, laptops, and wireless networks all provide access to the internal network that often bypasses the firewall. Intrusion detection systems may be effective at detecting suspicious activity, but do not provide *protection* against attacks. Recent worms such as Slammer and Blaster have such fast propagation speeds that by the time an alert is generated, the damage is done and spreading fast.

Intrusion Prevention Systems (IPS)

The inadequacies inherent in current defences has driven the development of a new breed of security products known as *Intrusion Prevention Systems* (IPS). This is a term which has provoked some controversy in the industry since some firewall and IDS vendors think it has been “hijacked” and used as a marketing term rather than as a description for any kind of new technology.

Whilst it is true that firewalls, routers, IDS devices and even AV gateways all have intrusion prevention technology included in some form, we believe that there are sufficient grounds to create a new market sector for true *Intrusion Prevention Systems*.

These systems are proactive defence mechanisms designed to detect malicious packets within normal network traffic (something that the current breed of firewalls do not actually do, for example) and stop intrusions dead, blocking the offending traffic automatically before it does any damage rather than simply raising an alert as, or after, the malicious payload has been delivered.

Within the IPS market place, there are two main categories of product: *Host IPS* and *Network IPS*, with the latter being further sub-divided into *Content-Based* and *Rate-Based* (or *Attack Mitigation*) systems.

Host IPS (HIPS)

As with Host IDS systems, the Host IPS relies on agents installed directly on the system being protected. It binds closely with the operating system kernel and services, monitoring and intercepting system calls to the kernel or APIs in order to prevent attacks as well as log them.

It may also monitor data streams and the environment specific to a particular application (file locations and Registry settings for a Web server, for example) in order to protect that application from generic attacks for which no “signature” yet exists.

One potential disadvantage with this approach is that, given the necessarily tight integration with the host operating system, future OS upgrades could cause problems.

Since a Host IPS agent intercepts all requests to the system it protects, it has certain prerequisites - it must be very reliable, must not negatively impact performance, and must not block legitimate traffic. Any HIPS that does not meet these minimum requirements should never be installed in a host, no matter how effectively it blocks attacks.

Network IPS (NIPS)

The Network IPS combines features of a standard IDS, an IPS and a firewall, and is sometimes known as an *In-line IDS* or *Gateway IDS (GIDS)*. The next-generation firewall - the *deep inspection firewall* - also exhibits a similar feature set, though we do not believe that the deep inspection firewall is ready for mainstream deployment just yet.

As with a typical firewall, the NIPS has at least two network interfaces, one designated as *internal* and one as *external*. As packets appear at either interface they are passed to the detection engine, at which point the IPS device functions much as any IDS would in determining whether or not the packet being examined poses a threat.

However, if it should detect malicious traffic, in addition to raising an alert, it will discard the packet(s) and mark that flow as bad. As the remaining packets that make up that particular TCP session arrive at the IPS device, they are discarded immediately.

Legitimate packets are passed through to the second interface and on to their intended destination. A useful side effect of some NIPS products is that as a matter of course - in fact as part of the initial detection process - they will provide “*packet scrubbing*” functionality to remove protocol inconsistencies resulting from varying interpretations of the TCP/IP specification (or intentional packet manipulation).

Thus any fragmented packets, out-of-order packets, or packets with overlapping IP fragments will be re-ordered and “cleaned up” before being passed to the destination host, and illegal packets can be dropped completely.

One thing to watch out for - don't let the "reactive" IDS vendors kid you into believing that they have *intrusion prevention* capabilities just because they can send TCP reset commands or re-configure a firewall when they detect an attack (a worrying piece of FUD that we have noticed in some IDS marketing literature recently).

The problem here is that unless the attacker is operating on a 2400 baud modem, the likelihood is that by the time the IDS has detected the offending packet, raised an alert, and transmitted the TCP Resets - and especially by the time the two ends of the connection have received the Reset packets and acted on them (or the firewall or router has had time to activate new rules to block the remainder of the flow) - the payload of the exploit has long since been delivered….. *game over*! Our guess is that there are not many crackers using 2400 baud modems these days….

A true IPS device, however, is sitting in-line - **all** the packets have to pass through it. Therefore, as soon as a suspicious packet has been detected - and **before** it is passed to the internal interface and on to the protected network, it can be dropped. Not only that, but now that flow has been flagged as suspicious, **all** subsequent packets that are part of that session can also be dropped with very little additional processing. Oh, and for good measure, some products are also capable of sending *TCP Resets* or *ICMP Unreachable* messages to the attacking host.

Rate-Based IPS (Attack Mitigator)

Most NIPS products are basically IDS engines that operate in-line, and are thus dependent on protocol analysis or signature matching to recognise malicious content within individual packets (or across groups of packets). These can be classed as *Content-Based IPS* systems.

There is, however, a second breed of Network IPS that ignores packet content almost completely, instead monitoring for anomalies in network traffic that might characterise a flood attempt, scan attempt, and so on. These devices are capable of monitoring traffic flows in order to determine what is considered "normal", and applying various techniques to determine when that traffic deviates from normal. This is not always as simple as watching for high-volumes of a specific type of traffic in a short space of time, since they must also be capable of detecting "stealth "attacks, such as low-rate connection floods and slow port scan attempts.

Since these devices are concerned more with anomalies in traffic flow than packet contents, they are classed as *Rate-Based IPS* systems - and are also known as *Attack Mitigators*, as they are so effective against DOS and DDOS attacks.

Detection Methods

At one time, most Network IDS/IPS products based their alerts purely on pattern matching packet contents against a database of known signatures. Then came a new breed of offerings that approached the problem in a completely different way - by doing a full protocol analysis on the data stream. Others began to use heuristics or anomaly-based analysis to determine when an attempted attack had taken place.

Today, most IDS/IPS employ a mixture of these detection methods in a single product, though some will be more biased towards one method than another.

According to Cisco, there are five main methods of attack identification (*source: Cisco Systems, The Science of Intrusion Detection System Attack Identification*):

Pattern Matching

Pattern matching in its most basic form is concerned with the identification of a fixed sequence of bytes in a single packet. In addition to the tell-tale byte sequence, most IPS will also match various combinations of the source and destination IP address or network, source and destination port or service, and the protocol. It is also often possible to tune the signature further by specifying a start and end point for inspection within the packet, or a particular combination of TCP flags.

The more specific these parameters can be, the less inspection needs to be carried out against each packet on the wire. However, this approach can make it more difficult for systems to deal with protocols that do not live on well defined ports and, in particular, Trojans, and their associated traffic, which can usually be moved at will.

Although it is often quite simple to define a signature for a particular exploit, basic pattern matching can often be too specific, sometimes requiring multiple signatures to be defined for minor variations in exploits. They are also prone to false positives, since legitimate traffic can often contain the relatively small set of criteria supposedly used to determine when an attack is taking place.

This method is usually limited to inspection of a single packet and, therefore, does not apply well to the stream-based nature of network traffic such as HTTP sessions. This limitation gives rise to easily implemented evasion techniques.

Stateful Pattern Matching

Stateful pattern matching offers a slightly more sophisticated approach, since it takes the context of the established session into account, rather than basing its analysis on a single packet.

Stateful IPS products must consider arrival order of packets in a TCP stream and should handle matching patterns across packet boundaries. Thus, if the exploit string to be matched is *foobar*, and the exploit is split across two packets, with *foo* in one and *bar* in another, the simple packet matching IPS will miss the attack, since it will not be able to match the complete string. The stateful IPS, however, will maintain the session context and reassemble the traffic stream, once again making the complete string available to the detection engine.

This requires more resources than simple pattern matching, since the IPS now has to allocate large amounts of memory and processing power to track a potentially large number of open sessions for as long as possible. This approach does make IPS evasion that much more difficult, though far from impossible.

Direction of traffic is also important here, both in terms of quality of detection and performance.

Client-to-server traffic inspection is the process of applying detection mechanisms to the “request side” portion of a communication - for example, in HTTP this could be the “GET” request coming from a client.

Client-to-server traffic inspection is typically activated to protect all traffic whether internally or externally generated. As the size of the traffic in terms of byte count is relatively small, the processing load placed on the IPS will be lower.

Server-to-client traffic inspection is the process of finding an attack in the "response side" portion of a communication - for example, in HTTP the server-to-client traffic could be the web page and content returned from the server as a result of a "GET" request. Server-to-client traffic, as in this example, is often much larger than the client-to-server traffic in terms of byte count. As a result, the processing load that is placed on an IPS is greater for server-to-client traffic.

Some vendors do not implement server-to-client signatures at all. Often this is for performance reasons, but sometimes it is a design decision by those vendors who also offer HIPS products, which are often better placed to detect the types of exploits executed by malicious response traffic as opposed to request traffic. Some vendors do include server-to-client signatures, but recommend they are disabled when performance is paramount. Bi-directional detection can have a significant impact on performance in some cases - those products which can handle this situation with zero or minimal impact on performance are worth closer inspection (although this level of performance often comes with a higher price tag).

It should be noted that there are situations where disabling server-to-client signatures is reasonably safe, and - happily - these are usually the situations where the highest levels of performance are demanded. Typically, this would be where an IPS is deployed within the network perimeter, where it is unlikely that purely internal HTTP response traffic is likely to be malicious. Perimeter defences would normally be deployed with both client-to-server and server-to-client signatures enabled, but perimeter devices rarely have the same performance requirements as internal ones.

Protocol Decode

Protocol decode IPS take a radically different approach to simple pattern matching IPS products - though sometimes not quite as radically different as the marketing folks would have you believe. With this technique, the IPS detection engine performs a full protocol analysis, decoding and processing the packet contents in the same way that the target client or server application would. It also tends to be stateful.

Although this may seem like using a sledgehammer to crack a nut, it does have the advantage of highlighting anomalies in packet contents much more quickly than doing an exhaustive search of a signature database. It also has the advantage of greater flexibility in capturing attacks that would be very difficult - if not impossible - to catch using pure pattern-matching techniques, as well as new variations of old attacks. These are attacks which - although changing only slightly from variant to variant - would normally require a new signature in the database for the "traditional" IPS architecture, but which would be detected automatically by a complete protocol analysis.

One of the first things the protocol decode engine does is to apply rules defined by the appropriate RFCs to look for violations. This can help to detect certain anomalies such as binary data in an HTTP request, or a suspiciously long piece of data where it should not be - a sign of a possible buffer overflow attempt.

One simple example of how this might work concerns searching Telnet login strings for one of the many well-known login names that rootkits tend to leave behind on the system. A pattern matching system might scan *all* Telnet traffic for *all* these patterns, in which case the more patterns you add, the slower it becomes (not *always* the case, but a reasonable assumption for the purposes of this example).

In contrast, a protocol analysis system will decode the Telnet protocol and extract the login name. It can then perform an efficient search in a binary-search tree or a hash table for just the login name, which should scale much better as new signatures are added.

In theory, therefore, protocol decoding should offer more efficient processing of traffic and improved scalability as more signatures are added, compared to a pure pattern matching solution. In reality, pattern matching solutions rarely opt for a “brute force” approach (there are some extremely intelligent and efficient pattern matching mechanisms available), and so the differences are not always as marked as the marketing people would like us to believe.

Note also, that pattern matching and protocol decoding are not mutually exclusive, as some would lead you to believe. A protocol analysis IPS can only go so far with its protocol decodes before it too will be forced to perform some kind of pattern matching, albeit against a theoretically smaller subset of “signatures”.

One major downside, of course, is that if a completely new type of exploit does surface, it is likely that the developer will have to create new protocol decode code to handle it, whereas the pattern matching approach can allow the administrator to develop a custom signature much more quickly on site.

Protocol decoding does offer a number of advantages, however. It minimises the chance for false positives if the protocol is well defined and enforced (although false positives can be higher if the RFC is ambiguous), and can also be more broad and general to allow the IPS to detect minor variations of an exploit without having to implement separate signatures.

You may see this technique referred to in several different ways:

- *Protocol decode*
- *Protocol Anomaly Detection*
- *Protocol validation*

Each of these terms, if strictly applied, could use a slightly different approach to the problem. For example, we would expect a *protocol decode* engine to perform the sort of additional pattern matching and length checking mentioned above on the field contents in order to detect specific exploits or buffer overflows.

Pure *protocol validation* or *Protocol Anomaly Detection* engines, however, might go no further than decoding just enough to be able to determine if the packet follows the RFC to the letter. If not, they will raise an alert - but in allowing a packet to pass, they cannot be sure that the contents will not contain a means of exploit that just happens to conform with the RFC.

Beware the marketing hype in this particular area – no matter what architecture is used, the performance figures and detection rates in a live deployment will speak for themselves.

Heuristic Analysis

Heuristic-based signatures use some kind of algorithmic logic on which to base their alarm decisions. These algorithms are often statistical evaluations of the type of traffic being presented.

A good example of this type of signature is one that would be used to detect a port sweep. This signature looks for the presence of a threshold number of unique ports being touched on a particular machine. The signature may further restrict itself through the specification of the types of packets that it is interested in (that is, SYN packets). Additionally, there may be a requirement that all the probes must originate from a single source, and even that valid SYN ACK packets must be seen to be returned by the host being probed.

Signatures of this type will react differently on different networks, and can be a significant source of false positives if not tuned correctly, requiring some threshold manipulations to make them conform to the utilisation patterns on the network they are monitoring. This type of signature may be used to look for very complex relationships as well as the simple statistical example given.

Anomaly Analysis

The final approach is to forget about trying to identify the attacks directly, and concentrate instead on ignoring everything that is considered “normal”. This is known as “*anomaly-based*” IPS, and the basic principle is that, having identified what could be considered “normal” traffic on a network, then anything that falls outside those bounds could be considered an “intrusion” - or at the very least, something worthy of note. This is generally better suited to passive IDS rather than in-line IPS devices, given its propensity for false positives.

The primary strength of anomaly detection is its ability to recognise previously unseen attacks, since it is no longer concerned with knowing what an attack looks like - merely with knowing what does not constitute normal traffic. Its drawbacks, of course, include the necessity of training the system to separate noise from natural changes in normal network traffic (the installation of a new - perfectly legitimate - application somewhere on the network, for example).

Changes in standard operations may cause false alarms while intrusive activities that appear to be normal may cause missed detections. It is also difficult for these systems to name types of attacks, and this technology has a long way to go before it could be considered ready for “prime time”.

Which Detection Method Is The Best?

Which detection method to choose is a difficult question, and in all honesty, it is not one with which most of those evaluating these products should concern themselves.

Adequate performance to handle the traffic to which the sensor will be exposed, accuracy of alerts, low incidence of false positives, and centralised management and reporting/analysis tools are far more important than how the packets are processed.

In some instances, the lines blur between methodologies to the point where they become almost indistinguishable.

For example, most protocol decode analysis engines alert the user to the presence of protocol violations that are not directly related to any known attack but are “anomalous” (for example, length-based buffer overflow detection). Therefore, in this instance the engine has attributes of an anomaly-based system.

As we have already mentioned, most protocol analysis systems are also reduced to performing some form of pattern-matching process following the protocol decode. Likewise, even the most basic pattern-matching systems perform some form of protocol analysis - even if it is only for a limited range of protocols. In truth, almost all Network IPS systems are already adopting a hybrid architecture.

By and large, therefore, the *pattern-matching vs. protocol decode* debate is one of religion - something for the marketing departments to shout about. Why should the average user care what happens under the hood as long as the product does what it claims to do - detect and prevent intrusions?

Implementation Challenges

There are a number of challenges to the implementation of an IPS device that do not have to be faced when deploying passive-mode IDS products. These challenges all stem from the fact that the IPS device is designed to work in-line, presenting a potential choke point and single point of failure.

If a passive IDS fails, the worst that can happen is that some attempted attacks may go undetected. If an in-line device fails, however, it can seriously impact the performance of the network.

Perhaps latency rises to unacceptable values, or perhaps the device fails closed, in which case you have a self-inflicted Denial of Service condition on your hands. On the bright side, there will be no attacks getting through! But that is of little consolation if none of your customers can reach your e-commerce site.

Even if the IPS device does not fail altogether, it still has the potential to act as a bottleneck, increasing latency and reducing throughput as it struggles to keep up with up to a Gigabit or more of network traffic. Devices using off-the-shelf hardware will certainly struggle to keep up with a heavily loaded Gigabit network, especially if there is a substantial signature set loaded, and this could be a major concern for both the network administrator - who could see his carefully crafted network response times go through the roof when a poorly designed IPS device is placed in-line - as well as the security administrator, who will have to fight tooth and nail to have the network administrator allow him to place this unknown quantity amongst his high performance routers and switches.

As an integral element of the network fabric, the Network IPS device must perform much like a network switch. It must meet stringent network performance and reliability requirements as a prerequisite to deployment, since very few customers are willing to sacrifice network performance and reliability for security. A NIPS that slows down traffic, stops good traffic, or crashes the network is of little use.

Dropped packets are also an issue, since if even one of those dropped packets is one of those used in the exploit data stream it is possible that the entire exploit could be missed.

Most high-end IPS vendors will get around this problem by using custom hardware, populated with advanced FPGAs and ASICs - indeed, it is necessary to design the product to operate as much as a switch as an intrusion detection and prevention device.

It is very difficult for any security administrator to be able to characterise the traffic on his network with a high degree of accuracy. What is the average bandwidth? What are the peaks? Is the traffic mainly one protocol or a mix? What is the average packet size and level of new connections established every second - both critical parameters that can have detrimental effects on some IDS/IPS engines? If your IPS hardware is operating “on the edge”, all of these are questions that need to be answered as accurately as possible in order to prevent performance degradation.

Another potential problem is the good old *false positive*. The bane of the security administrator’s life (apart from the script kiddie, of course!), the false positive rears its ugly head when an exploit signature is not crafted carefully enough, such that legitimate traffic can cause it to fire accidentally. Whilst merely annoying in a passive IDS device, consuming time and effort on the part of the security administrator, the results can be far more serious and far reaching in an in-line IPS appliance.

Once again, the result is a self-inflicted Denial of Service condition, as the IPS device first drops the “offending” packet, and then potentially blocks the entire data flow from the suspected hacker. If the traffic that triggered the false positive alert was part of a customer order, you can bet that the customer will not wait around for long as his entire session is torn down and all subsequent attempts to reconnect to your e-commerce site (if he decides to bother retrying at all, that is) are blocked by the well-meaning IPS.

Another potential problem with any Gigabit IPS/IDS product is, by its very nature and capabilities, the amount of alert data it is likely to generate. On such a busy network, how many alerts will be generated in one working day? Or even one hour? Even with relatively low alert rates of ten per second, you are talking about 36,000 alerts every hour. That is 864,000 alerts each and every day. The ability to tune the signature set accurately is essential in order to keep the number of alerts to an absolute minimum. Once the alerts have been raised, however, it then becomes essential to be able to process them effectively. Advanced alert handling and forensic analysis capabilities - including detailed exploit information and the ability to examine packet contents and data streams - can make or break a Gigabit IDS/IPS product.

Of course, one point in favour of IPS when compared with IDS is that because it is designed to prevent the attacks rather than just detect and log them, the burden of examining and investigating the alerts - and especially the problem of rectifying damage done by successful exploits - is reduced considerably.

Requirements for effective prevention

Having pointed out the potential pitfalls facing anyone deploying these devices, what features are we looking for that will help us to avoid such problems?

- **In-line operation -** only by operating in-line can an IPS device perform true protection, discarding all suspect packets immediately and blocking the remainder of that flow

- **Reliability and availability -** should an in-line device fail, it has the potential to close a vital network path and thus, once again, cause a DoS condition. An extremely low failure rate is thus very important in order to maximise up-time, and if the worst should happen, the device should provide the option to fail open or support fail-over to another sensor operating in a fail-over group (see below). In addition, to reduce downtime for signature and protocol coverage updates, an IPS must support the ability to receive these updates without requiring a device re-boot. When operating inline, sensors rebooting across the enterprise effectively translate into network downtime for the duration of the reboot
- **Resilience -** as mentioned above, the very minimum that an IPS device should offer in the way of High Availability is to fail open in the case of system failure or power loss (some environments may prefer this default condition to be “fail closed” as with a typical firewall, however - the most flexible products will allow this to be user-configurable). Active-Active stateful fail-over with cooperating in-line sensors in a fail-over group will ensure that the IPS device does not become a single point of failure in a critical network deployment
- **Low latency -** when a device is placed in-line, it is essential that its impact on overall network performance is minimal. Packets should be processed quickly enough such that the overall latency of the device is as close as possible to that offered by a layer 2/3 device such as a switch, and no more than a typical layer 4 device such as a firewall or load-balancer.
- **High performance -** packet processing rates must be at the rated speed of the device under real-life traffic conditions, and the device must meet the stated performance with all signatures enabled. Headroom should be built into the performance capabilities to enable the device to handle any increases in size of signature packs that may occur over the next three years. Ideally, the detection engine should be designed in such a way that the number “signatures” (or “checks”) loaded does not affect the overall performance of the device.
- **Unquestionable detection accuracy -** it is imperative that the quality of the signatures is beyond question, since false positives can lead to a Denial of Service condition. The user MUST be able to trust that the IDS is blocking only the user selected malicious traffic. New signatures should be made available on a regular basis, and applying them should be quick (applied to all sensors in one operation via a central console) and seamless (no sensor reboot required)
- **Fine-grained granularity and control -** fine grained granularity is required in terms of deciding exactly which malicious traffic is blocked. The ability to specify traffic to be blocked by attack, by policy, or right down to individual host level is vital. In addition, it may be necessary to only alert on suspicious traffic for further analysis and investigation
- **Advanced alert handling and forensic analysis capabilities -** once the alerts have been raised at the sensor and passed to a central console, someone has to examine them, correlate them where necessary, investigate them, and eventually decide on an action. The capabilities offered by the console in terms of alert viewing (real time and historic) and reporting are key in determining the effectiveness of the IPS product.

The NSS Intrusion Prevention Group Test

The NSS Group conducted the first comprehensive IPS test of its kind, now updated in this Edition.

This exhaustive review will give readers a complete perspective of the capabilities, maturity and suitability of the products tested for their particular needs.

As part of its extensive IPS/Attack Mitigator test methodologies (see section on *Testing Methodology* later in this report for detailed methodologies, updated for this latest test) The NSS Group subjects each product to a brutal battery of tests that verify the stability and performance of each IPS tested, determine the accuracy of its security coverage, and ensure that the device will not block legitimate traffic.

If a particular IPS has been designated as *NSS Approved*, customers can be confident that the device will not significantly impact network/host performance, cause network/host crashes, or otherwise block legitimate traffic.

To assess the complex matrix of IPS/Attack Mitigator performance and security requirements, the NSS Group has developed a specialised lab environment that is able to exercise every facet of an IPS product. The test suite contains over 800 individual tests that evaluate IPS products in three main areas: *performance and reliability, security accuracy,* and *usability*.

This thorough review should give readers a complete perspective of the capabilities, maturity and suitability of the products tested for their particular needs.

Performance

Any IPS is expected to be reliable (not crash), to never block legitimate traffic, and to not unduly affect network or host system performance.

The latency and throughput of a Network IPS (NIPS) or Attack Mitigation device must be on a par with other equipment in the network on which it is deployed, and in this respect, an in-line NIPS must strive to perform much more like a switch than a typical passive security device, especially when it is necessary to install more than one NIPS in the same data path.

Detection/Blocking Performance Under Load

This group of tests verifies that the IPS does not adversely impact legitimate traffic, even when new TCP connections are being created rapidly. We also verify that the sensor is capable of detecting and blocking exploits when subjected to increasing loads of background traffic up to the maximum bandwidth supported as claimed by the vendor. An IPS that misses attacks under load can be evaded. An IPS that adversely affects legitimate background traffic will not stay in-line for long.

A fixed number of exploits are launched with zero background traffic to ensure the sensor is capable of detecting our baseline attacks. Once that has been established, increasing levels of varying types of background traffic are generated **through** the IPS device in order to determine the point at which the sensor begins to miss attacks.

All tests are repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic (or up to the maximum rated throughput of the device in 25 per cent increments should this be less than 1Gbps). The test is conducted with UDP, HTTP, and mixed-protocol traffic and includes packet rates up to 453,000 packets per second and connection rates up to 20,000 connections per second.

Latency & User Response Times

In any network environment latency is important. Latency may impose an upper bound on throughput and it also has an impact on interactive applications, thus affecting user response time. As such, it is important to understand the impact of latency introduced by a NIPS and to determine the maximum acceptable delay, which will be different for each network.

There is a direct relationship between latency introduced by a networking device and the maximum throughput allowed by that device on a single TCP connection. There is a critical value for the *round trip time* (RTT) of a packet in each network, and if the latency is below this critical value, TCP throughput will be unaffected - instead, it is the line speed of the underlying network which becomes the bottleneck. Above this critical value, however, TCP throughput is negatively impacted. To be specific, the maximum throughput achievable for any given TCP connection in a zero loss network is expressed as:

throughput = window / RTT

where *window* is the maximum TCP window size (64 Kbytes by default) and RTT is the round trip time in the network.

This equation tells us that the throughput of a TCP connection is inversely proportional to network latency (note that this is TCP throughput for *one* connection - the aggregate bandwidth is not affected by latency). In other words, if you double latency, you halve throughput.

Consider adding a NIPS in an internal Gigabit network where the RTT is 200 microseconds. The critical value for RTT in a Gigabit network is 500 microseconds (below which it may no longer be possible to achieve 1Gbps of throughput), which means the NIPS can add a maximum of 300 microseconds to the RTT without affecting the network. In this particular case, therefore, for an internal, high speed deployment, the administrator may determine that his chosen IPS device needs to be capable of sub-300 microsecond latency under normal traffic loads.

Of course, the latency of an IPS device may vary significantly based on packet size, complexity of the protocol, presence of attack traffic, or simply the makeup of the normal traffic passing through it. For example, Gigabit segments, will rarely carry only a single TCP connection. Rather, a saturated Gigabit segment could be supporting hundreds, if not thousands of TCP connections, and this multiplexing eases the impact of latency on the overall throughput on the segment.

Although each of these connections carries only a fraction of the total throughput, a few connections tend to dominate. The maximum latency for a NIPS is then determined by the utilisation of the fastest connection. For example, in a Gigabit Ethernet segment carrying 10,000 TCP connections the fastest connection might have a throughput of 250Mbps. In this case, the critical value for round trip latency is as high as 2 milliseconds.

Assuming the latency without the NIPS is 300 microseconds, an administrator may therefore determine that his chosen NIPS device must be capable of 1700 microsecond round trip latency (850 microseconds in each direction).

Such critical value calculations are important when TCP connections achieve maximum throughput, which is true for large data transfers.

For smaller data transfers, and non-TCP applications like NFS, latency has a more direct impact on user experience - response time is directly proportional to latency. That is, *doubling latency doubles response time.* In these situations, the latency of the network in which a NIPS is deployed determines the acceptable latency of the NIPS.

Consider deploying a hypothetical NIPS with 1 millisecond one-way latency in the following scenarios:

- In internal corporate LANs, the round trip latency could be in the 200-300 microsecond range. Deploying our hypothetical NIPS would increase the maximum round trip latency to 2.3 milliseconds, an increase of just over 700 per cent. The time to copy a large group of files, for example, would increase by a factor of seven.
- In inter-campus corporate networks connected over a MAN, the latency could be in the 500-1000 microsecond range (or less). Deploying our hypothetical NIPS would increase the maximum round trip latency to 3 milliseconds, a minimum increase of 300 per cent. The time to copy a large group of files, for example, would increase by at least factor of three.
- Internet facing connections experience round-trip latency from 10-100 milliseconds. Deploying our hypothetical NIPS would increase the round trip latency by 1-10 per cent, which would have only a minor impact on the user experience.

The latency of the NIPS must therefore be evaluated in the context of the network in which it is deployed. For example, to protect networks that are accessed over the public Internet, one-way NIPS latencies in the 1-2 millisecond range would be acceptable. Whereas for NIPS deployments on MAN/WAN links, NIPS latencies of well under 1 millisecond would be essential. And as we have already mentioned, for deployments on internal networks where latencies are a few hundred microseconds, NIPS latencies of less than 300 microseconds would be more appropriate.

Network administrators have laboured long and hard to reduce latency within the corporate network to an absolute minimum. Core network devices such as switches are frequently chosen as much on their performance - packet loss and latency under all load conditions - as any other feature. Given that Network IPS devices are operating in-line, it is not surprising that they will be evaluated in a similar way.

For this reason, part of The NSS Group methodology uses very similar testing techniques to those we would normally employ when testing switches (in order to determine *packet latency*), in **addition** to measuring *application latency*. This group of tests determine the effect the IPS sensor has on the traffic passing through it under various load conditions. High packet latency will lower TCP throughput. High application latency will create a negative user experience.

Bi-directional network latency of a range of differently-sized UDP packets is measured under three test conditions: with no load, with 500 Mbps of HTTP traffic (or half the rated load of the device if this is less than 1Gbps), and while the device is under a heavy SYN flood attack (up to 10 per cent of the rated throughput of the sensor).

Spirent Avalanche and Reflector devices are also used to generate HTTP sessions through the device in order to gauge how any increases in latency will impact the user experience in terms of failed connections and increased Web response times.

This "*application latency*" is measured both with no background load and while the device is under attack.

Stability & Reliability

These tests verify the stability of the IPS device under various extreme conditions. Long-term stability is critical for an in-line IPS device, where failure can produce network outages.

In the first part of this test, we expose the external interface of the sensor to a constant stream of attacks over an extended period of time. The device is configured to block and alert, and thus this test provides an indication of the effectiveness of both the blocking and alert handling mechanisms. A continuous stream of exploits mixed with some legitimate sessions is transmitted through the sensor at a maximum rate of 90 per cent of the claimed throughput of the device for eight hours with no additional background traffic.

The device is expected to remain operational and stable throughout this test, blocking 100 per cent of recognisable exploits, raising an alert for each, and passing 100 per cent of legitimate traffic. If any recognisable exploits are passed - caused by either the volume of traffic or the IPS device failing open for any reason - this will result in a FAIL. If any legitimate traffic is blocked - caused by either the volume of traffic or the IPS device failing closed for any reason - this will also result in a FAIL.

In the second part of the test we stress the protocol stack of the device under test by exposing it to malformed traffic from the ISIC test tool for eight hours. The device is expected to remain operational and capable of detecting and blocking exploits throughout the test to attain a PASS.

We scan the management interface for open ports and active services and report on known vulnerabilities. We also stress the protocol stack of the management interface of the NIPS by exposing it to malformed traffic from the ISIC test tool. The device is expected to remain (a) operational and capable of detecting and blocking exploits, and (b) capable of communicating in both directions with the management server/console throughout the test to attain a PASS. We also note whether the sensor detects the ISIC attacks even though targeted at the management port.

Security Effectiveness

Detection Accuracy & Breadth

This group of tests verifies that the NIPS will not block legitimate traffic (*Accuracy*) and is capable of detecting and blocking a wide range of common exploits (*Breadth*). Although *breadth* is extremely important, *accuracy* is critical because a NIPS that blocks legitimate traffic will not remain in-line for long.

We have a number of trace files of normal traffic with "suspicious" content, together with several "neutered" exploits that have been rendered completely ineffective. The IPS attains a "PASS" for each test case if it does **not** raise an alert and does **not** block the traffic. Whilst it is not possible to validate completely the entire signature set of any IPS, this test demonstrates how accurately the IPS detects and blocks a wide range of common exploits, port scans, and Denial of Service attempts.

This test is repeated twice: the first run with blocking disabled on the IPS in order to determine which attacks are detected and how accurately they are detected (*Attack Recognition Rating*); the second run with blocking enabled in order to determine which attacks are blocked successfully regardless of how they are detected or what alerts are raised (*Attack Blocking Rating*).

Following the initial test run, each vendor is provided with a list of CVE references of the attacks missed and is allowed 48 hours to produce an updated signature set. This updated signature set must be released to the general public as a standard signature/product update before the report is published - this ensures that vendors do not attempt to code signatures just for this test.

Naturally, Rate-Based IPS devices will not respond to the same attack traffic as Content-Based devices, and so for those the Detection Accuracy tests involve detecting and mitigating a wide range of rate-based attacks such as port scans, SYN floods, connection floods, and so on. We note which of these are mitigated completely, which are mitigated partially, and which require the use of built-in firewall capabilities.

Resistance To Evasion Techniques

These tests verify that the IPS is capable of detecting and blocking basic exploits when subjected to varying common evasion techniques. An IPS that cannot detect attacks subjected to these “script kiddie” evasion techniques is easily bypassed.

The tests consist of four parts (only the third is applicable to Rate-Based devices):

- ***Baselines -*** *This establishes that the IPS is capable of detecting and blocking a number of common basic attacks (our baseline suite) in their normal state, with no evasion techniques applied.*
- ***Packet Fragmentation and Stream Segmentation -*** *The baseline HTTP attacks are repeated, running them through fragroute using 19 evasion techniques.*
- ***URL Obfuscation -*** *The baseline HTTP attacks are repeated, this time applying 9 URL obfuscation techniques made popular by the Whisker Web server vulnerability scanner.*
- ***Miscellaneous Evasion Techniques -*** *Certain baseline attacks are repeated, and are subjected to 7 protocol- or exploit-specific evasion techniques, including altering default ports, inserting spaces in FTP command lines, inserting non-text Telnet opcodes in FTP data streams, and RPC record fragging.*

For each of the evasion techniques, we note if (i) the attempted attack is blocked successfully (the primary aim of any IPS device), (ii) the attempted attack is detected and an alert raised in **any** form, and (iii) if the exploit is successfully “decoded” to provide an accurate alert relating to the original exploit, rather than alerting purely on anomalous traffic detected as a result of the evasion technique itself.

Stateful Operation

If the IPS is tracking TCP session state, then it has the potential to introduce denial of service when the session table becomes full (too many connections) or if it can’t keep up with the creation of new sessions (too many connections per second).

As with latency and bandwidth, the number of connections supported by the IPS and its connection per second rate should be matched to the network.

For example, a fully saturated Gigabit Ethernet link can handle 22,000 5KByte transfers per second. Assuming each connection lasts 20 seconds, the IPS should be able to handle 448,000 simultaneous connections. These numbers scale proportionately for slower networks. Any IPS that doesn't offer these capabilities will impact performance of Web or e-commerce servers.

The aim of this section is to be able to determine whether the IPS is capable of monitoring stateful sessions established through the device at various traffic loads without either losing state or incorrectly inferring state.

An IPS that does not maintain TCP session state can flood the management console with false-positive alerts. Although this should not directly impact the IPS blocking function, it can make it very hard to perform forensic analysis of the attacks. In addition, if the default condition of the sensor is to block all traffic for which it does not believe there is a current connection in place, then an inability to maintain state under extreme conditions could result in the sensor blocking legitimate traffic by mistake.

In the first part of this test, we transmit a number of packets taken from capture files of valid exploits, but without first establishing a valid session with the target server. In order to receive a "PASS" in this test, no alerts should be raised for any of the actual exploits. However, each packet should be blocked if possible since it represents a "broken" or "incomplete" session.

In part two, we test whether the sensor is capable of preserving state across increasing numbers of open connections, as well as continuing to detect and block new exploits while not blocking legitimate traffic when the state tables are filled. Various numbers of TCP sessions from 10,000 to 1,000,000 (one million) are tested.

This test is run in both the out-of-box configuration and then repeated after applying any tuning recommended by the vendor (if applicable) to increase the size of the state tables.

Usability

After quantitatively evaluating the network performance and security effectiveness of the IPS, we qualitatively evaluate the features and usability of the product.

This evaluation provides the reader with valuable insight into product features, how easy it is to install the IPS and perform common, day-to-day operations with the management console. Areas evaluated include *installation, configuration, policy editing, alert handling,* and *reporting and analysis*.

JUNIPER NETWORKS IDP 600F V3.1

Executive Summary

The Juniper Networks Intrusion Detection and Prevention (IDP) system is a turnkey appliance-based system which uses as many as eight detection methods to detect malicious network traffic.

The IDP 600F is capable of operating in both in-line mode (as an *Intrusion Prevention System*) or as a passive *Intrusion Detection System* attached to a SPAN or mirror port on a switch.

The IDP 600F is designed for 500Mbps networks and can handle its maximum rated bandwidth under most normal traffic conditions likely to be encountered on a sub-Gigabit network. On a typical network, we would deem the 500Mbps rating to be conservative. Latency is excellent under all conditions and all packet sizes providing the device is not subjected to heavy DOS/DDOS attacks. We recommend that the device is deployed behind an effective attack mitigation appliance.

We found the IDP 600F to be to be very stable and reliable, coping with our extensive reliability tests with ease and without blocking any legitimate traffic or succumbing to common evasion techniques.

The management system has been well designed to handle management and configuration of large numbers of sensors across the enterprise. Policy definition and deployment is extremely flexible and powerful, and the alert handling and reporting/forensic capabilities are extensive - some of the most flexible we have seen in terms of drill-down and quick reporting capabilities.

Architecture

IDP uses a three-tier architecture that consists of the *IDP Sensor*, the *IDP Management Server*, and the *IDP User Interface*.

- ***IDP Sensor -*** *monitors the network on which the IDP system is installed. The Sensor is a purpose-built hardware appliance that runs the IDP Sensor software.*
- ***IDP Management Server -*** *stores and manages all Attack Objects (including attack signatures and protocol anomalies), log information, rulebases, and Protection Policies. Multiple Sensors can be managed by a single Management Server.*
- ***User Interface (UI) -*** *a Java-based graphical interface for interacting with the IDP system. The UI is used to access remotely and manipulate the information stored on the Management Server.*

IDP Sensor

With this latest release, Juniper has replaced the old standard Dell hardware with a purpose-built appliance of its own, available in four main versions:

- ***IDP 50 -*** *A 1U rack mount appliance with two 10/100/1000Mbps copper ports for monitoring, and one for management. The IDP 50 contains 1GB of RAM, and supports a maximum of 50Mbps throughput*

- ***IDP 200 -*** *A 2U rack mount appliance with eight 10/100/1000Mbps copper ports for monitoring, one for management and one for dedicated HA operation. The IDP 200 contains 1GB of RAM, and supports a maximum of 250Mbps throughput. Redundant power supplies are an option.*
- ***IDP 600C/600F -*** *A 2U rack mount appliance with ten Gigabit monitoring ports (ten 10/100/1000Mbps copper ports in the model C, and eight Gigabit fibre ports plus two 10/100/1000Mbps copper ports in the model F). Two dedicated 10/100/1000Mbps ports are also provided, one for management and one for HA operation. The IDP 600 contains 4GB of RAM, and supports a maximum of 500Mbps throughput. Redundant power supplies and RAID disk array are included.*
- ***IDP 1100C/1100F -*** *A 2U rack mount appliance with ten Gigabit monitoring ports (ten 10/100/1000Mbps copper ports in the model C, and eight Gigabit fibre ports plus two 10/100/1000Mbps copper ports in the model F). Two dedicated 10/100/1000Mbps ports are also provided, one for management and one for HA operation. The IDP 1100 contains 4GB of RAM, and supports a maximum of 1Gbps throughput. Redundant power supplies and RAID disk array are included.*

The model submitted for testing was the IDP 600F, a 2U device with Intel Xeon processor rated at 500Mbps, and sporting a total of ten monitoring interfaces (eight fibre and two copper) allowing it to monitor up to five in-line pairs or ten subnets in passive mode. The Sensor's primary task is to detect suspicious and anomalous network traffic based on specific rules defined in IDP rulebases. If the Sensor is running in-line, it can also take a predefined action against malicious traffic.

IDP Sensors can be deployed as *active gateways* or *passive sniffers*. A *passive sniffer* Sensor connects to a switch or hub in promiscuous mode and sniffs the network traffic as it passes by. The Sensor monitors network traffic, records security events, and can create alarms for attacks. However, because a sniffer Sensor cannot take direct action against the attack, it cannot prevent it.

An *active gateway* Sensor sits between the network and a firewall, or a network and a DMZ, and takes an active role in protecting the network. When it detects intrusions or attacks defined by Protection Policies, the Sensor can drop, block, or ignore the suspicious connection, or drop only the suspicious packets. Because active gateway Sensors are installed in-line they can take direct action against the attack, thus preventing it.

There are four in-line deployment options:

- ***Transparent and Bridge Modes:*** *In this mode, the IDP is transparent and no reconfiguration needs to be done to let the hosts know that it is there. Moreover, the hosts are not aware that the IDP is even on the network. For the IDP system, the transparent mode is the easiest to deploy and is the recommended mode of operation. Using third party load balancing solutions, IDP can be installed in a High Availability configuration in Bridge and Transparent mode. The device under test was deployed in Transparent mode.*
- ***Proxy-ARP Mode:*** *In this mode, the host machines are aware that the IDP is on the network, and IDP automatically reconfigures the host machines to send it all packets that need to be forwarded. Proxy-ARP mode typically has higher throughput performance than Bridge/Transparent Modes, but only works with networks where all traffic is routed through a single router. The IDP can be installed in Standalone High Availability configuration in Proxy-ARP mode.*

- ***Router Mode:*** *In this mode, IDP behaves as a typical router, using a routing table to determine where the packets need to be sent and then forwarding the packets to the correct destination. In a typical network, the IDP will thus be configured as the default gateway for all the protected hosts. Router mode is traditionally associated with older security devices, such as Firewalls, and is only offered in IDP for compatibility. The IDP can be installed in a High Availability configuration using third party load balancing solutions or in a standalone HA configuration in Router mode.*

The Sensor stores logs on the local hard drive initially, and then transmits the logs to the Management Server. If communication between the Sensor and Management Server fails, the Sensor attempts to restore communication and stores new logs on the local file system until available disk space is consumed. If no disk space remains and communication has not been restored, the Sensor first deletes sysinfo logs (debug and error messages) followed by packet captures to free up disk space. If communications cannot be restored before all local disk space is consumed, the Sensor ceases to detect new attacks.

Detection Engine

IDP operates as an in-line, active device that inspects all traffic and determines what constitutes an intrusion. IDP detection mechanisms can be applied bi-directionally to detect attacks in both client-to-server and server-to-client traffic (as described in the report introduction).

If instructed by the Security Policy, IDP can deny traffic associated with the intrusion by dropping the connection or associated packets (when operating in-line). Rules are created in the IDP rulebases to specify what actions IDP takes when a particular attack is detected. Detection methods (rulebases) include:

- ***Stateful Signature -*** *IDP includes several hundred signature Attack Objects that use stateful signatures to detect known attacks. The administrator can view, create, edit, or delete signature Attack Objects using the Signature Editor in the UI. Compound signatures allow multiple individual signatures (both stateful signatures and protocol anomalies) to be combined into a single compound signature to detect complex attacks in a single session.*

  The language for creating custom signatures includes Perl style regular expressions to make it easier to create signatures that can be used to detect an underlying vulnerability rather than a specific exploit based on that vulnerability. The number of stateful signature fields available when writing custom signatures is over 500.

- ***Protocol Anomaly -*** *Protocol anomalies are deviations from the standard protocol. Most legitimate traffic adheres to the published specifications (RFC) for protocols, but traffic that doesn't produces an anomaly. The IDP system uses the RFC to create protocol anomaly Attack Objects for deviations from the protocol's published specification. The latest release has added a further eight protocols, bringing the total number of decoded protocols to almost 60.*

- ***Backdoor -*** *Backdoor detection uses network traffic patterns and heuristics of packet transmissions to detect interactive traffic, a common sign of an attacker using a Trojan or backdoor. Unlike anti-virus software, which scans for* ***known*** *backdoor files or executables on the host system, IDP detects the interactive traffic that is produced when backdoors are used, but without requiring a specific signature.*

*This method ensures that IDP can detect all backdoors, both known **and** unknown, even if the data is encrypted. Normal bi-directional services (such as FTP) are excluded from the default policies.*

- ***Traffic Anomaly -*** *A traffic anomaly is a pattern that indicates abnormal network activity such as port scans and other distributed network attacks. The IDP system counts the number of ports scanned in a specified time period and uses this traffic flow analysis to identify scans.*
- ***IP Spoofing -*** *IP spoofing occurs when a packet uses a fake IP address, and the IDP system detects IP spoofing by comparing the IP addresses of packets to the IP addresses of devices on the protected network.*

  An IP address is considered spoofed if an incoming packet uses an IP address that belongs to a Network Object on the internal network, or if an outgoing packet uses an IP address that does not belong to a Network Object on the internal network
- ***DOS Detection -*** *IDP detects and prevents DOS attacks such as SYN floods by ensuring that the TCP handshake is performed correctly. In in-line mode, SYN requests can be proxied by IDP until they are completed successfully. Any incomplete SYN requests can be discarded by IDP after time without affecting the host at which the original request was aimed.*
- ***Layer 2 Attack Detection -*** *Attackers can manipulate Layer 2 protocols to perform ARP attacks (such as ARP cache poisoning) and other MAC attacks. The IDP system detects Layer 2 attacks by defining implied rules on the IDP Sensor. Implied rules include ARP table restrictions, fragment-handling, connection timeouts, byte/length thresholds for data packets, and other mechanisms.*
- ***Network Honeypot -*** *Whilst not strictly speaking a detection method, the IDP Network Honeypot impersonates open ports on existing servers in order to lure attackers into attempting exploits for the purpose of evidence gathering.*

High Availability

When operating in-line, it is essential that traffic is not interrupted by device failures. It is therefore possible to deploy the IDP system (IDP 200 and above) in a *high availability* (HA) configuration to provide failure protection, either in a standalone configuration or using third-party hardware.

In high availability, two to sixteen Sensors are joined together in a HA cluster that provides failure protection and/or load balancing. In *load balancing* mode, all Sensors in the cluster share network traffic equally, and if a Sensor fails, network traffic is redirected to the other Sensors in the cluster. With *hot standby*, a primary Sensor handles all network traffic while a secondary Sensor stands by. If the primary Sensor fails, network traffic is redirected to the secondary Sensor.

State information is shared between Sensors in the cluster using dedicated state-sync network interfaces. High availability configurations are available for transparent, bridge, router, and proxy-ARP deployment modes.

An integrated hardware bypass is included for all copper detection ports on all models, allowing fail-open operation in the event of failure. Third party bypass units are required to provide the same capability on fibre ports.

IDP Management Server

The IDP *Management Server* consists of software that manages system resources and data that drives IDP functionality. The IDP Management Server requires an enterprise class server running Sun Solaris 8/9 (SPARC) or Linux RedHat 7.2, 8 or RHEL 3.0 (Intel). It centralises the logging, reporting, data, and Security Policy management for the IDP system. Logs from multiple Sensors are consolidated into a single, central repository, and the Management Server also provides a single point of management, policy definition and deployment. All objects, Security Policies, and logs are stored in databases on the Management Server and are administered using the User Interface by one or more administrators.

Management Server communication with the other two tiers of the IDP system (the Sensor and User Interface) is encrypted and authenticated to provide an additional level of security. The Management Server gathers log records for security events using a proprietary communication mechanism utilising a modified version of UDP which Juniper Networks claims is more reliable than ordinary UDP, requires less CPU and memory resources from servers, and reduces the number of acknowledgement packets on the network. Alerts are stored in a proprietary database which Juniper Networks also claims offers high levels of performance (over 20,000 events per second).

When IDP detects a DoS attack, the Sensor increases its transmission rate and injects redundant packets to increase the odds of transmitting the logs to the central location.

User Interface (UI)

The User Interface (UI) is software that provides a graphical environment for centrally managing IDP. The UI is a java-based software application that can be installed on multiple computers across the network.

Although the UI supports multiple users, only one user at a time can take control of the Management Server - this eliminates concerns about synchronisation or data loss. Each administrator can configure the UI with his own preferences - IDP stores user preferences and custom Log Viewer views in the central database so that they remain consistent when accessed from different client machines.

Performance

The aim of this section is to verify that the sensor is capable of detecting and blocking exploits when subjected to increasing loads of background traffic up to the maximum bandwidth supported as claimed by the vendor.

For each type of background traffic, we also determine the maximum load the IPS can sustain before it begins to drop packets/miss alerts. It is worth noting that devices which demonstrate 100 per cent blocking but less than 100 per cent detection in these tests will be prone to blocking **legitimate** traffic under similar loads.

The IDP 600F was tested as a 500Mbps device, and performance at almost all levels of our load tests was excellent, with 100 per cent of all attacks being detected and blocked under most load conditions. The one exception was Test 4.2.4 where we determined that there was a limit of approximately 8,000 HTTP connections per second (equating to approximately 400Mbps) meaning the test could not be completed.

Beyond this limit, legitimate connections began to fail, but all attack traffic was still blocked successfully.

However, we would happily confirm Juniper's 500Mbps rating for this device under normal network conditions, and would actually consider it to be conservative on a typical network.

Basic latency figures were very good for a device of this type at all traffic loads and packet sizes, ranging from 86µs with 125Mbps of 256 byte packets, to 145µs with 500Mbps of 1000 byte packets. Behaviour through all of the tests with no background traffic was very predictable, with relatively small increases in latency as traffic levels increased from 125Mbps to 500Mbps across each packet size.

Placing the device under a half load of 250Mbps of HTTP traffic we noted small increases in latency across the board, rising from 86µs to 133µs with 250 byte packets, from 111µs to 149µs with 550 byte packets, and from 138µs to 175µs with 1000 byte packets.

These results are excellent for a 500Mbps device, and given that HTTP response times were also excellent, we believe that the IDP 600F could be deployed anywhere on a 500Mbps network, either internally or at the perimeter.

Whether or not the SYN Protector was enabled, the IDP 600F was unable to handle the 50Mbps of SYN flood traffic we launched during our DOS/DDOS tests. Once we had exceeded the 8000 connections per second (i.e. 8000 SYNs per second) maximum we discovered in the HTTP stress tests, the device became unresponsive, effectively blocking all traffic, both malicious and legitimate.

More effective SYN protection is planned for a future release, but for now we recommend the IDP 600F is deployed behind an attack mitigation device or firewall with full DOS/DDOS mitigation capabilities.

The IDP 600F performed consistently and completely reliably throughout our tests. Under eight hours of extended attack (comprising millions of exploits mixed with genuine traffic) it continued to block 100 per cent of attack traffic, whilst passing 100 per cent of legitimate traffic.

Exposing the sensor interface to ISIC-generated traffic had no adverse effect, and the device continued to detect and block all other exploits (as well as raising ISIC-related alerts) throughout and following the ISIC attack.

Please refer to the *Testing Methodology* section for full details of the methodology used and performance results.

Security Effectiveness

We installed one IDP 600F sensor with the latest signature set. The Security Policy was created from scratch, switching on all Attack Objects including *Critical*, *High, Medium* and *Low* (omitting only *Informational* signatures, which are more suited to auditing purposes), as well as turning on *Backdoor Protection* and the *SYN Protector*.

We also enabled fragmented packet detection and flow error detection in order to identify certain DOS and ICMP attacks.

Finally, we disabled all HTTP server-to-client signatures for performance reasons. These would not generally be required for a typical internal deployment, where performance would be paramount. There is a severe performance impact when enabling HTTP server-to-client signatures, reducing throughput of the device to 200-250Mbps and making it more suitable for a perimeter deployment.

Signature recognition (with blocking disabled) was excellent out of the box at 95 per cent. This figure was increased to 97 per cent following a signature pack update which Juniper provided within 48 hours. Blocking performance was identical throughout the tests. Note that with the HTTP server-to-client signatures enabled, the attack recognition rose to a perfect 100 per cent (though with a corresponding decrease in maximum performance as a result).

All of our “false negative” (modified exploit) cases were detected correctly out of the box, demonstrating that the Juniper signatures are generally designed to detect the underlying vulnerability rather than a specific exploit.

A major concern in deploying an IPS is the blocking of legitimate traffic, and the IDP 600F did block three of our false positive test cases out of the box. This was rectified following the signature update, after which we noted no further false positives throughout the rest of the testing.

The IDP 600F comes with a number of suggested Protection Policy templates to aid the administrator when setting up initially. These include suggested policies for file servers, Web servers, DNS servers, and a typical “default” policy, and are based on a number of carefully constructed Dynamic Groups which will allow Juniper (or the administrator) to modify these policies automatically with each signature update.

Resistance to known evasion techniques was excellent, with the IDP 600F achieving a clean sweep across the board in all our main evasion tests. *Fragroute, Whisker, ADMmutate* and even *RPC record fragging* all failed to trick the IDP 600F into ignoring valid attacks.

Not only were the fragmented and obfuscated attacks blocked successfully, but all but one of them - a single *fragroute* evasion - were decoded accurately as well when not blocking. Only a couple of miscellaneous FTP evasion techniques were effective in evading the device - although we do not consider this to be too serious, it is a situation which we would prefer to see remedied as soon as possible, and Juniper is working on a solution at the time of writing.

Out of the box, the device maintained state on up to 100,000 open connections, successfully detecting our half-open exploit as it was completed. It also continued to detect and block new exploits as we maintained 100,000 open connections, and no legitimate traffic was blocked during this stage of these tests.

With a simple change to the flow-related parameters in the GUI, it was possible to maintain state across 500,000 open connections (though Juniper actually recommends 220,000 as the maximum on this device).

Default operation of the device is to refuse new connections when the state tables are full or resources are low, meaning that legitimate traffic is blocked and new attacks are not alerted. However, state is maintained completely on existing flows and attack traffic should never be allowed through the device.

Stateless “exploits” are not alerted upon (this is correct behaviour in order to be resistant to *Stick* and *Snot* tools), and mid-flows are blocked by default. It is not possible to configure the device to allow mid-flows.

Please refer to the *Testing Methodology* section for full details of the methodology used and performance results.

Usability

This part of the test procedure consists of a subjective evaluation of the features and capabilities of the product, and covers *installation, configuration, policy editing, alert handling,* and *reporting and analysis*.

Installation

Installation is very straightforward. The Management Server can be installed on any *secure and trusted* Sun Solaris 8/9 (SPARC) or Linux RedHat 7.2/8 or RHEL 3.0 (Intel) host. It is installed via a simple shell script that walks the administrator through all the steps necessary to install the software, and this process takes no more than five minutes.

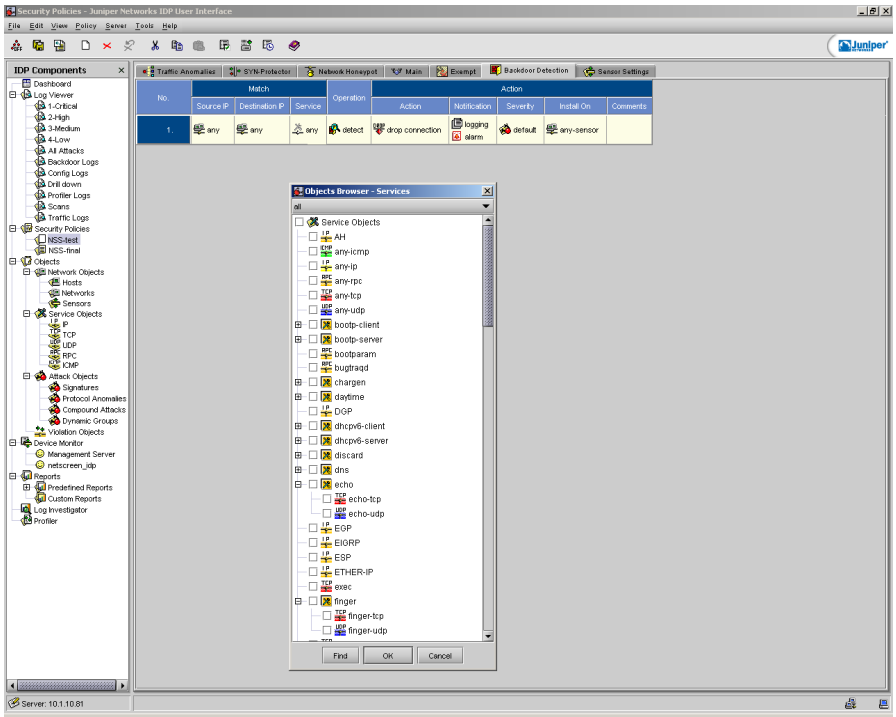

Figure 1 - IDP 600F: The UI

Next, the Sensor is connected to the network to be protected, and one of the interfaces connected to the management network. Using the *Appliance Configuration Manager* (ACM), a Web-based tool, a Wizard guides the administrator through the six-section configuration process of configuring the Sensor software. During this process, key parameters such as Sensor name, passwords, deployment mode, network interface settings, SSH settings, date and time are configured.

The administrator is also prompted to enter a One Time Password (OTP) and is provided with a unique identification number (VIN) for the Sensor. This is used later when adding Sensor devices in the UI.

Finally, the User Interface (UI) is installed - versions are available for Windows and Red Hat Linux. Once network components have been added (starting with the Sensors themselves) and a Security Policy defined and deployed (see following section), the IDP 600F is ready to go.

Configuration

The rule-based, centralised management system makes it easy to configure, update, and maintain the IDP system from a single administration point if required. An unlimited number of IDP Sensors can be managed with a single, logical Security Policy thanks to the ability to apply individual rules to specific sensors within the same policy. Once the administrator has defined an enterprise-wide Security Policy it can be distributed at the push of a button. All relevant configuration and signature information is automatically sent to the IDP Sensors in a secure manner (authenticated and encrypted).

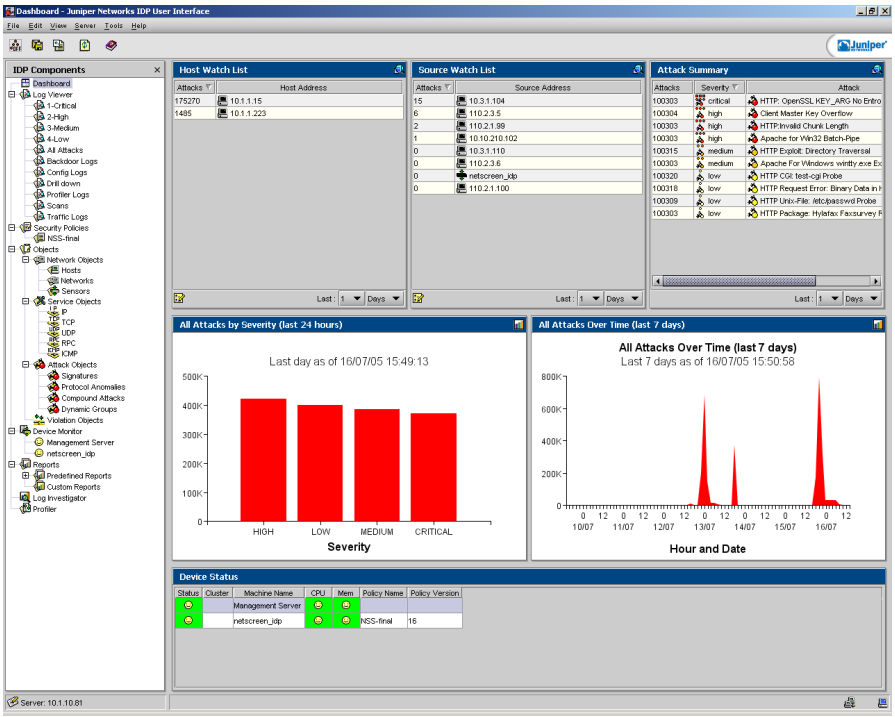

Figure 2 - IDP 600F: The Dashboard display

The UI contains eight main components:

- *Dashboard*
- *Log Viewer*
- *Security Policy Editor*
- *Object Editor*
- *Device Monitor*
- *Reports*
- *Log Investigator*
- *Profiler*

Each of the above options is accessed via a hierarchical menu tree in a pane on the left of the UI (some of the above options have several sub-levels within the menu tree), whilst data is displayed in a second pane on the right. The Dashboard component is the default component shown in the main display area when the administrator first logs in.

This provides a one-screen high-level view which displays the vital statistics of the network and the IDP system (though it can be slow to refresh when the number of alerts is excessive).

The data that appears in the Dashboard is real time, and refreshes periodically from the IDP Management Server. It is possible to drill-down quickly from the high-level display to the detailed events behind it. The administrator can customise the Dashboard to display only the information that is most important to him, including:

- ***Target Host Watch List -*** *to monitor target hosts and/or networks of particular interest*
- ***Source Watch List -*** *to monitor activity from particular source addresses*
- ***Attack Summary -*** *list of the most recent/common attacks*
- ***Reports -*** *Any one of the built-in reports can be regularly refreshed on the Dashboard, making it very customisable and of more relevance to individual administrators*
- ***Device Status -*** *To monitor the status of the Management Servers and all Sensors (availability, CPU utilisation, disk space, etc.). Thresholds can be set to flag devices as requiring attention if critical parameters are exceeded (i.e. CPU utilisation pegged above 90%, or disk space below a certain value).*

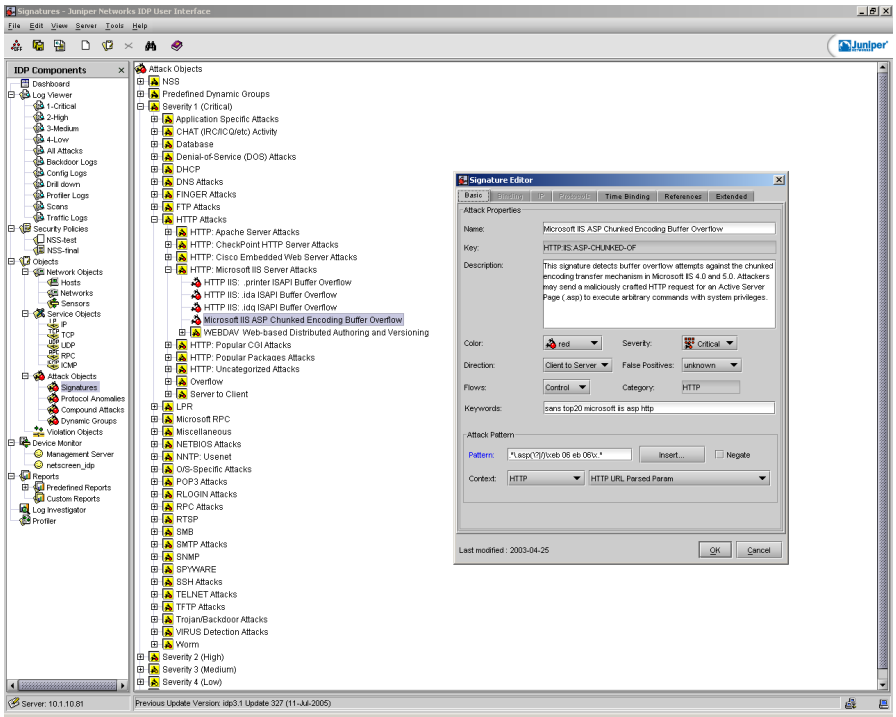

Figure 3 - IDP 600F: Browsing Attack Objects

The first step in making the IDP system functional is to create *Network Objects* for each of the IDP Sensors. *Objects* are the IDP representation of the components of the system, including:

- ***Network Objects -*** *representing network components (individual hosts, networks, servers, Sensors).*
- ***Service Objects -*** *representing services running on the network (such as FTP, HTTP, and Telnet)*

- ***Attack Objects -*** *representing attack signatures and protocol anomalies, used to create rules for Security Policies.*
- ***Violation Objects -*** *allows the administrator to define which hosts and networks are allowed communicate, and on which ports. The Profiler will then report on any violations to these rules, enabling the enforcement of a corporate policy which goes beyond pure security.*

Network Objects are the basic building blocks used in Security Policy rules to specify both the network components and the Sensor(s) which will protect those components.

Attack Objects are used in IDP rules to detect attack signatures and protocol anomalies that could compromise the network being protected. IDP includes a database of several hundred Attack Objects that are updated daily - a simple update procedure allows the administrator to connect to the Juniper Networks Web site or to upload Objects from a file which has been previously downloaded and stored locally.

The update procedure provides a detailed report of which Objects are new, which are updated, and which will be deleted (a nice touch, and one we would like to see more often in other products) - custom amendments can be preserved if required. The administrator has the option to select all the suggested changes or only some of them as required - this is a very well thought out update procedure.

Policy Management

Attack Objects are categorised and grouped according to severity first (*Critical, High, Medium, Low* or *Informational*), and target application or platform second.

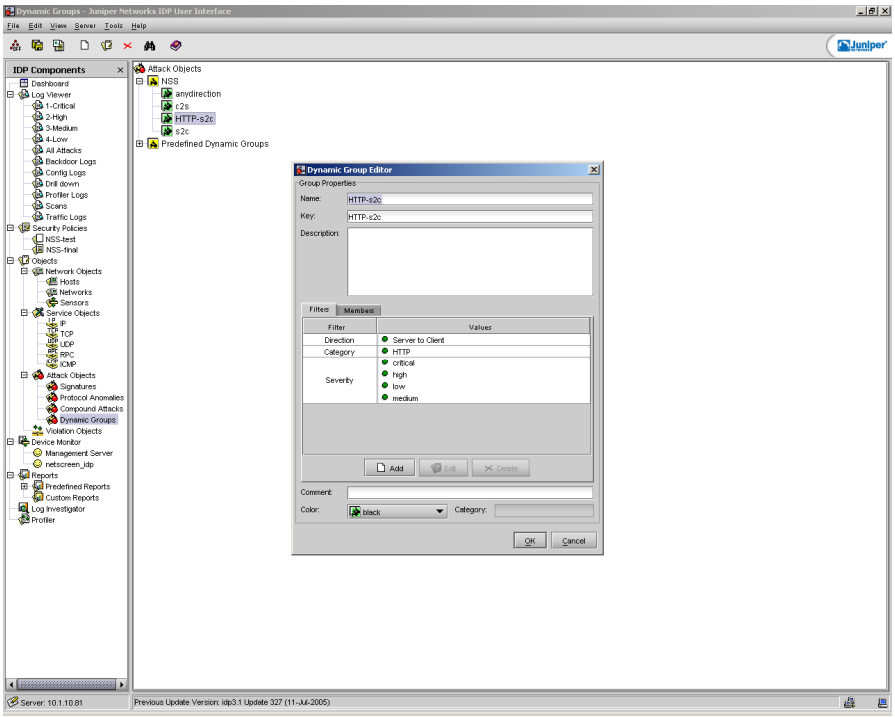

Figure 4 - IDP 600F: Configuring dynamic groups

This is useful in that it allows the administrator to create a policy which, for example, monitors only for *Critical* exploits.

The concept of *dynamic groups* of signatures allows the administrator to create groups which will be updated automatically as new signatures are added to the product. Dynamic groups make it possible to categorise signatures by various properties including vendor (i.e. all Microsoft applications), application (i.e. all IIS or Apache signatures), protocol (i.e. SMB only), direction (i.e. client to server), severity (i.e. critical only) and potential for false positives (i.e. never).

These filters can be combined in many different ways, allowing the administrator to build very specific groups (i.e. all critical SMB signatures that are client-to-server and have a low possibility for false positives) or wide-ranging ones (i.e. all Microsoft applications). Whenever signature updates are downloaded, the appropriate signatures are added to the correct dynamic groups automatically, ensuring, for example, that your IIS policy automatically includes all the latest IIS signatures with no intervention required. This is an excellent feature, and is used extensively by Juniper to create some very useful default Policy templates.

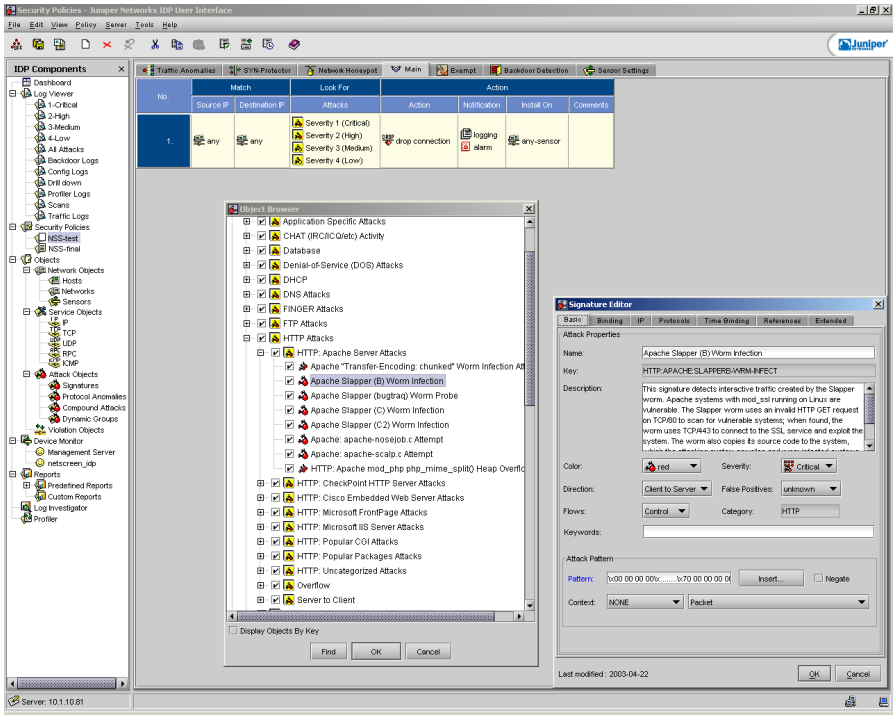

Figure 5 - IDP 600F: Policy Editor

The Object Editor provides the means to view and edit pre-defined Objects, as well as create new ones (including custom signatures).

When defining Security Policies in the IDP UI, they take the form of multiple *rules* laid out in a format that will be very familiar to most firewall administrators. Rules are organised into *rulebases*, collections of rules that use the same detection method to detect and prevent network intrusions. There are seven IDP rulebases which combine to form a Security Policy:

- *Traffic Anomaly*
- *SYN-Protector*
- *Network Honeypot*
- *Main (Attack Objects)*
- *Backdoor Detection*
- *Sensor Settings (sensor-specific parameters)*
- *Exempt (to allow certain traffic to be ignored)*

Each rulebase uses a different detection mechanism to protect against intrusions (see *Architecture* section for details).

It is possible to have any number of Security Policies stored in the IDP system (several Policy Templates are provided to help kick-start the definition process), but each IDP Sensor can use only one Security Policy at a time. Installing a Security Policy on an IDP Sensor gives the Sensor a set of instructions about the actions the Sensor should take in response to specific network situations. The same Security Policy can be installed on multiple IDP Sensors, or unique Security Policies can be installed on each Sensor in the network.

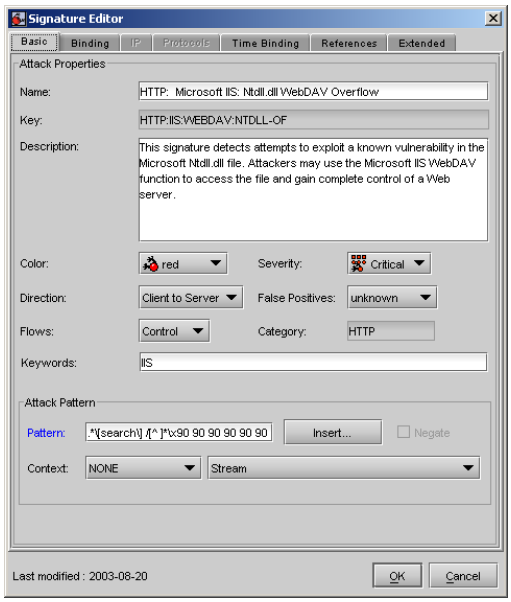

Figure 6 - IDP 600F: Defining custom signatures

When creating custom rules, the administrator defines the patterns or circumstances to be matched against each packet (including protocol, source and destination IP addresses, source and destination ports, flow direction, context within packet, packet contents, and so on), and the actions taken when a match is found.

Each Attack Object has seven tabs of information which can be accessed via the Signature Editor:

- *The **Basic tab** specifies the attack properties and the signature (attack pattern) that IDP uses to match attacks. This tab also includes the context in which the attack pattern occurs.*
- *The **Binding tab** specifies the service that the attack uses to attack the network.*
- *The **IP tab** specifies the packet fields, values, and options in a malicious packet.*
- *The **Protocols tab** specifies data length, flags, and other detailed data for the protocol header of a malicious packet.*
- *The **Time Binding tab** specifies time-related thresholds (such as how many suspicious packets within a specific time interval are required before the traffic is considered malicious)*
- *The **References tab** specifies known network security references for this attack.*
- *The **Extended Information** tab that provides detailed information from the TruSecure IntelliShield database*

The creation of new pattern-matching signatures is relatively straightforward, although a knowledge of regular expressions would be a requirement in most cases for specification of the exploit patterns to be matched. Protocol anomaly signatures can neither be edited nor created from scratch, since these require extensive coding and can thus only be added or modified by Juniper Networks developers.

However, IDP does include the ability to define *Compound Signatures*, which allow multiple individual signatures (both stateful and protocol anomaly) to be combined into a single signature to enable detection of complex exploits. This is an extremely powerful and flexible capability, providing for the creation of some very complex and accurate custom signatures.

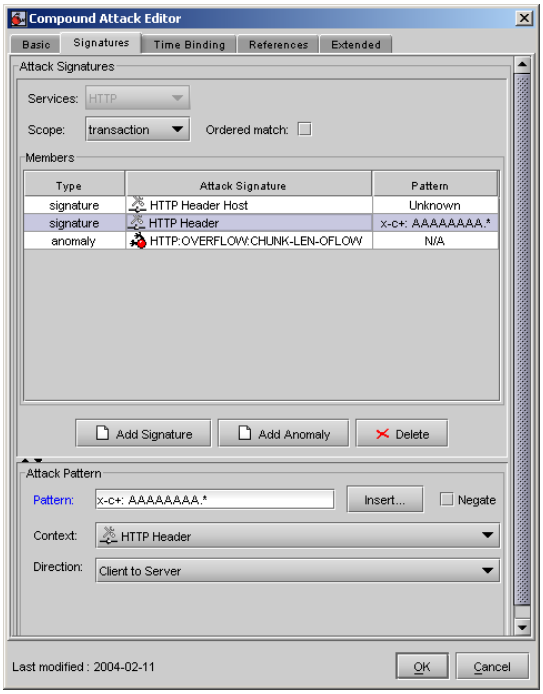

Figure 7 - IDP 600F: Defining Compound Signatures

IDP is one of the more open systems currently available, providing the means to copy and edit the built-in Attack signatures. All of the critical fields from the protocol decoders are exposed to the UI, and can be used in regular expressions to match for suspicious traffic.

Note that this is more than simply “pattern matching”, where a signature has to look for a particular pattern in a lengthy data stream. Instead, it is possible for an individual field - say the FTP user name - to be isolated and used in complex expressions. These can then be combined together (perhaps looking for a particular user name followed by a password greater than a specified length) into the aforementioned Compound Signatures.

The IDP contains a default set of actions that the Sensor can perform against network traffic that matches the specified pattern or circumstance when operating in-line. These actions include :

- ***None -*** *takes no action against the connection.*
- ***Ignore -*** *ignores the remainder of a connection after a Attack Object in a Security Policy rulebase is matched.*

- ***Drop packet -*** *drops a matching packet before it can reach its destination but does not close the connection. Used to drop packets for attacks in traffic that is prone to spoofing, such as UDP traffic, where the dropping of a connection for such traffic could result in a denial of service that prevents subsequent receipt of traffic from a legitimate source IP address.*
- ***Drop connection -*** *drops the connection without sending a RST packet to the sender, preventing the traffic from reaching its destination. Used to drop connections for traffic that is not prone to spoofing.*
- ***Close client and server -*** *closes the connection and sends a RST packet to both the client and the server. If the IDP Sensor is in sniffer mode, IDP sends a RST packet to both the client and server but does NOT close the connection.*
- ***Close client -*** *closes the connection to the client, but not to the server.*
- ***Close server -*** *closes the connection to the server, but not to the client.*

If the current network traffic matches a rule, IDP can perform an IP action against future network traffic that uses the same IP address. IP actions are similar to other actions in that they tell IDP to drop or close the connection, but they also allow the Sensor to block the attacker for a specified amount of time.

A number of notification options are also available whenever an attempted exploit is detected:

- ***Log -*** *Generate a log record for security events*
- ***Alarm -*** *If alarm is selected and the rule is matched, IDP places an alarm flag in the alarm column of the Log Viewer for the matching log record (used primarily for rapid filtering of alerts with higher than normal levels of importance)*
- ***Tracking Session Data -*** *Session data includes the number of packets passed, the number of bytes, and the elapsed time. Detailed session data can be tracked in order to enter the information into a log or traffic analysis tool, monitor user activity on the network, determine the length of sessions, or monitor the number of bytes transferred*
- ***SNMP Trap -*** *Sends an SNMP trap to the SNMP Manager.*
- ***Syslog Message -*** *Sends an entry to a syslog server.*
- ***E-mail -*** *Sends an e-mail to designated recipients.*
- ***Run Script -*** *Run custom scripts that take information about a security event and process it in a specific way.*
- ***Logging Packets -*** *It is possible to record the individual packets in the network traffic that matched a rule by capturing the packet data for the attack. By default, only the trigger packet is captured, though it is also possible to specify a number of packets to be captured both before and after the trigger - a nice feature.*

Action and Alert options can only be set at the individual rule level. Given that a single rule could contain a single Attack Object or every single Attack Object available, it could be said that this is as flexible as it needs to be.

However, in order to improve readability, most Security Policies would consist of multiple rules. The ability to set Action and Alert options on a global Policy-wide basis would thus be a welcome addition, saving the administrator the effort of having to change these options for every single rule in a Policy if, for example, he wanted to turn on or off packet logging for all Objects.

When editing Policies, it is often useful to be able to search for existing signatures that fit specific criteria. The IDP UI contains a search facility which allows the administrator to search for one or more signatures based on various criteria including name, key words, description, CVE reference, BugTraq ID, false positive rating or target application/platform. This is available at both the Attack Object level and within the Policy Editor (the latter being a very welcome new capability in the V3.0 release).

It does, however, only find signatures one at a time, requiring you to keep pressing “next” in order to find all matches for a given search. It also allows you to search on only a single parameter at a time. It would be nice if it allowed the specification of multiple parameters in a single search and returned multiple signatures in the result set, perhaps then allowing the application of bulk edits to the entire group.

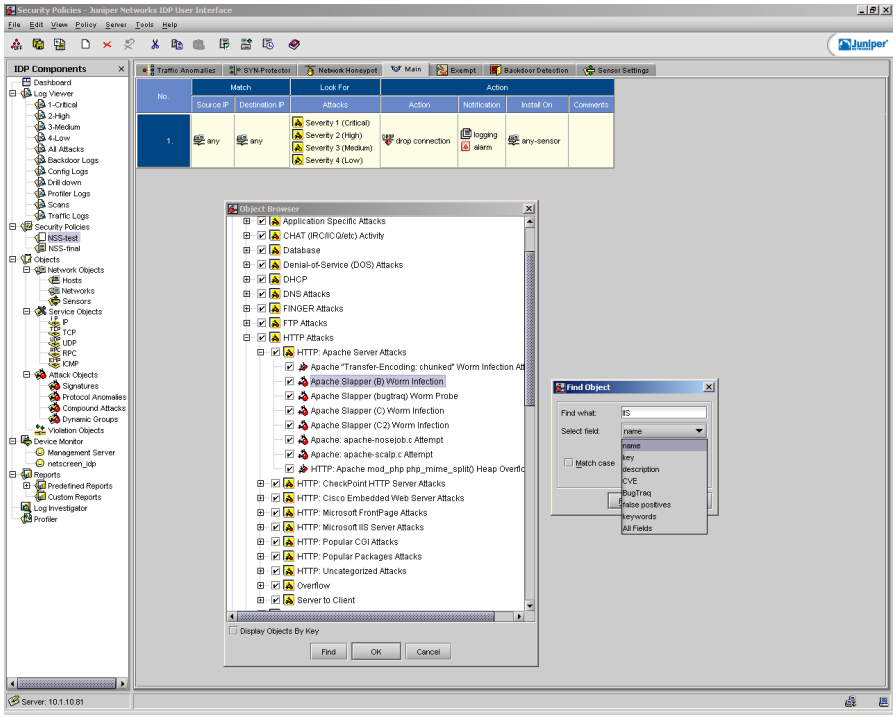

Figure 8 - IDP 600F: Searching for Objects within Policy Editor

The *Exempt* rulebase allows the administrator to specify that certain legitimate traffic be ignored for the purposes of matching for exploits. For example, vulnerability scanners can trigger many signatures, and so it would be possible to define rules that allowed a specific machine to scan one or more servers on the corporate network without hindrance. These rules can be defined manually, or by selecting an alert and clicking on the “*Exempt*” menu option. The latter method is ideal for rapidly fine tuning a security policy based on false alarms which may have been raised.

Finally, it is worth noting that all of the main Sensor operational parameters - such as flow table sizes, flow timeouts, fragmentation reassembly parameters, maximum flows, thresholds, etc. - are exposed in the *Settings* tab in the Policy Editor. Thus it is possible to alter default values to accommodate environments where, for example, it is necessary to monitor more than the default 100,000 open connections. The default values are adequate for most circumstances, but it is nice to see parameters such as these made available for editing in such a straightforward manner.

Once a Security Policy has been created and verified, it can be deployed to one or more Sensors - a list of Sensors appears during deployment allowing the administrator to select all Sensors (the default), groups of Sensors or an individual Sensor. One useful addition would be the ability to logically group Sensors together in order to select a specific collection of Sensors (those protecting DMZ networks, for example) with a single check box.

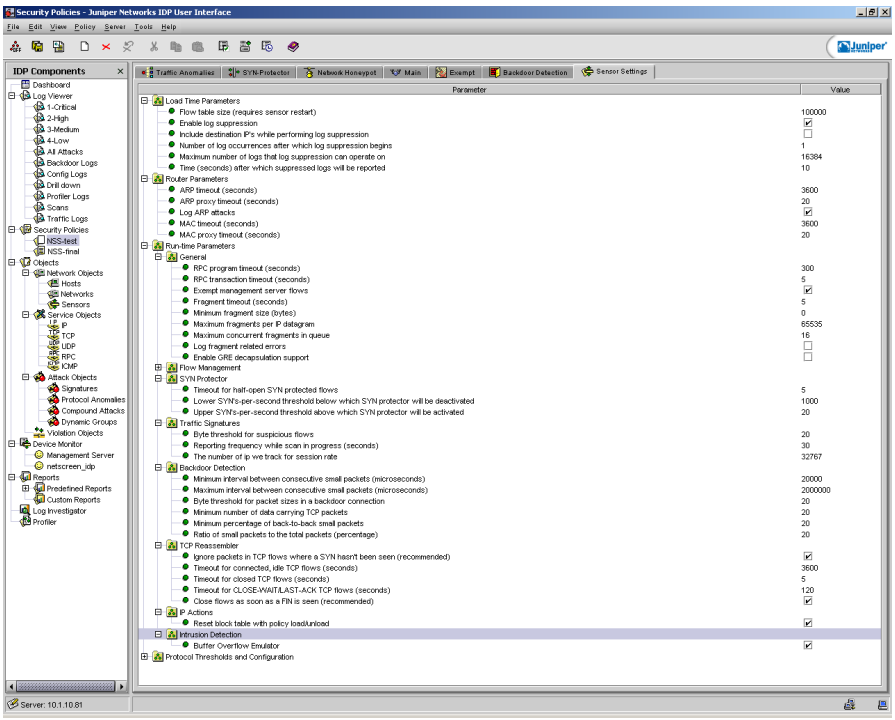

Figure 9 - IDP 600F: Sensor settings

When creating or editing individual rules within a Policy, it is possible to specify whether that rule applies to all Sensors or one or more individual devices. When the Security Policy is subsequently deployed, the rules become active only on the Sensors selected in the rulebase. This provides the option to have a single global Security Policy, with individual rules activated or deactivated on a per-Sensor basis. Alternatively, it is also possible to create multiple Security Policies to be applied to different Sensors.

This is an extremely flexible method of handling Policy creation and deployment, and is a feature we would like to see implemented in more IDS products.

A Sensor can only use one Security Policy at a time, and when a new Security Policy is installed, it overwrites any existing policies on the Sensor. Each time Policy is deployed, however, the IDP saves a version of the Policy currently installed. A version is a copy of the original Security Policy, and contains information about when the policy was installed on the Sensors and which user performed the install.

It is possible to view, reinstall, and delete previous versions of a Security Policy. However, because Policy versions are designed to provide an audit trail of the rules that were operating on a Sensor at a given time, it is not possible to edit previously installed Security Policy versions. The administrator can print and export (as PDF or postscript) all rulebases in a Security Policy.

Alert Handling

In the Log Viewer, the administrator can view log records that IDP Sensors generate based on criteria defined in the Security Policies.

The Log Viewer displays log records in table format and can be modified using filters or column settings. When viewing log entries it is important to be able quickly to access all information relevant to that particular alert.

Thus, IDP provides one-click navigation between the Log Viewer (shows incidents), the Session Viewer (shows associated packet contents in detail), the Signature Editor (shows why the incident was triggered), the Security Policy Editor (shows which Policy triggered the event), the *Enterprise Security Profiler* (ESP) (provides an on-demand view of network activity for detailed analysis), and the *Security Explorer* (provides a means to focus on a particular host, showing all activity relating to it specifically).

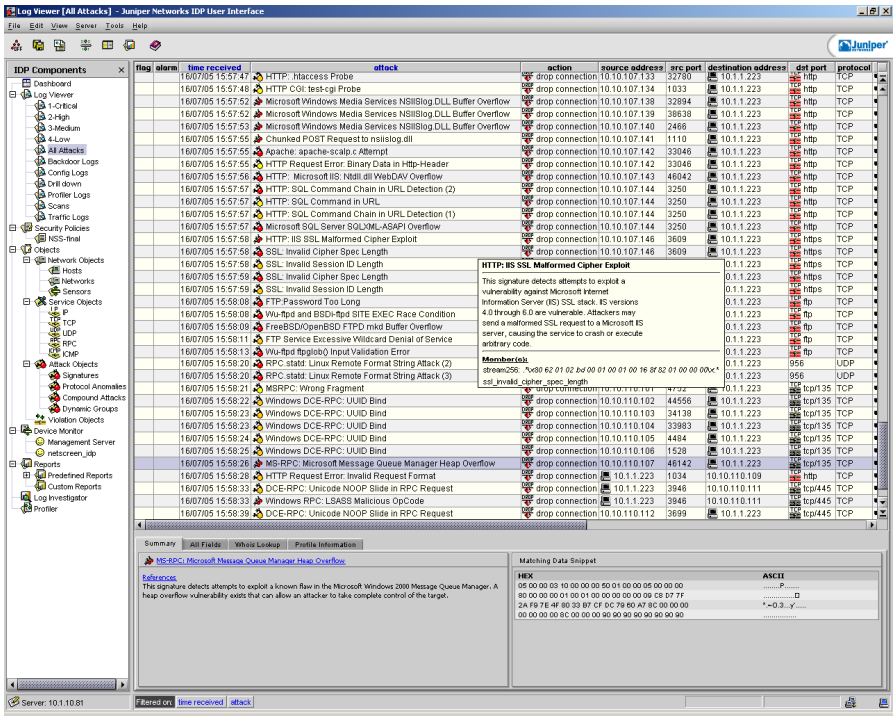

Figure 10 - IDP 600F: Log Viewer

The *Signature Editor* also contains a *References* tab that provides links to outside information about the attack, such as the attack CVE number and description, as well as an *Extended Information* tab that provides detailed vulnerability and exploit information from the TruSecure IntelliShield database.

The ability to access directly the Security Policy that triggered a particular alert is extremely useful when attempting to refine Policies to reduce the incidence of false positives. If an alert is identified as a false positive, the administrator can access the Policy with a single click, disable or tune that Attack Object, and re-deploy the Policy immediately.

Log records contain a great deal of data about the activity on the network, but it is not always necessary nor desirable to see all the information at one time.

The administrator can use the Log Viewer to manipulate, analyse, and export the information contained in the log records to display only the information that is of immediate interest. For example, using the Log Viewer it is possible to:

- *View complete, summarised, or detailed information for each log record*
- *Show, hide, or move columns, and filter data by column headings*
- *Create a new instance of the Log Viewer that shows only custom filters and column settings - these can be saved as custom views*
- *Set flags on log records to indicate a specific priority or action*
- *Launch the Packet Viewer*
- *Print or export log record data*

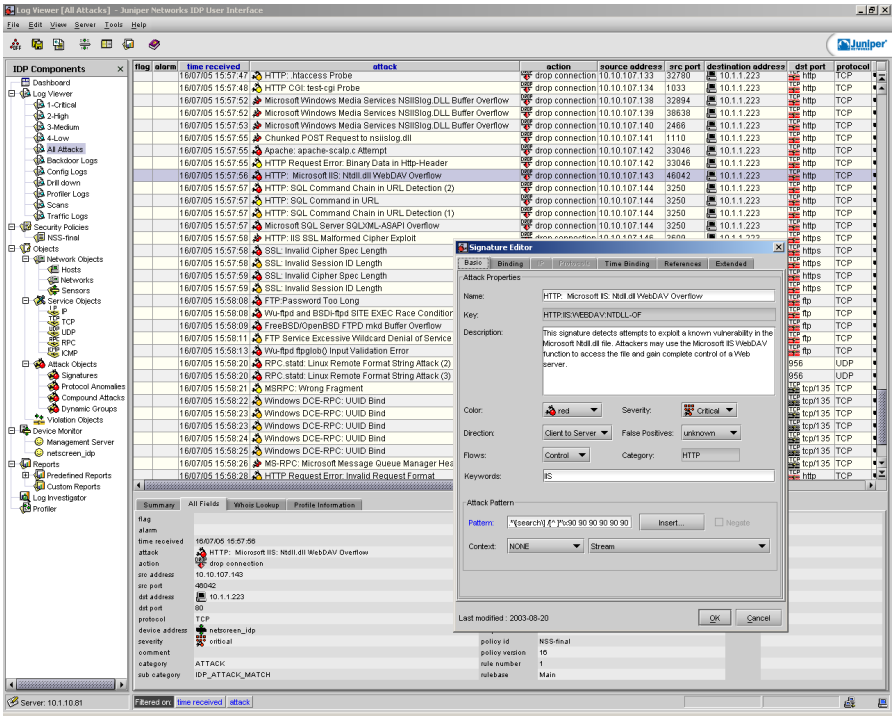

Figure 11 - IDP 600F: Viewing all alert fields plus signature details via the Log Viewer

Although the main part of the Log Viewer screen provides a tabular view of the log entries, selecting individual entries populates four tabs at the bottom of the screen which provide more detail on the alert in question.

The "*All Fields*" tab duplicates all the data from the log record in a condensed format removing the need to scroll across and thus making it easier to read. The "*Summary*" tab provides a detailed description of the security event (including extensive material from the TruSecure database), and includes links to external references (such as the CVE database).

The remaining two tabs at the foot of the screen provide an instant *whois* lookup for the selected IP address, and access to detailed host, application and networking information collected by the *Enterprise Security Profiler* (ESP) module (covered in more detail in the *Reporting and Analysis* section).

Right-clicking on any log entry brings up a sub-menu which allows the administrator immediately to:

- *Set filters based on that entry - this includes or excludes all other similar entries in order to provide a more focussed list of alert entries*
- *Search for similar alerts - this finds similar entries one after the other each time the "next" button is pressed, rather than grouping them together for display as would happen when setting a filter*
- *Display packet data, security policy or attack/signature data*
- *Run Quick Reports - this produces rapid reports based on search criteria populated automatically from the selected alert. For example, by right clicking on the destination address a Quick Report can be launched which instantly provides the recent Critical Severity attacks to that particular address. This can be further configured (i.e. to display Top 10 attacks to that address) and then saved as Custom Reports to be re-run on demand.*
- *Flag the alert.*

A number of flags can be set to aid other administrators, including the priority of an alert, and whether it: is a false positive; has been assigned to an investigator; should be followed up; is pending; is closed.

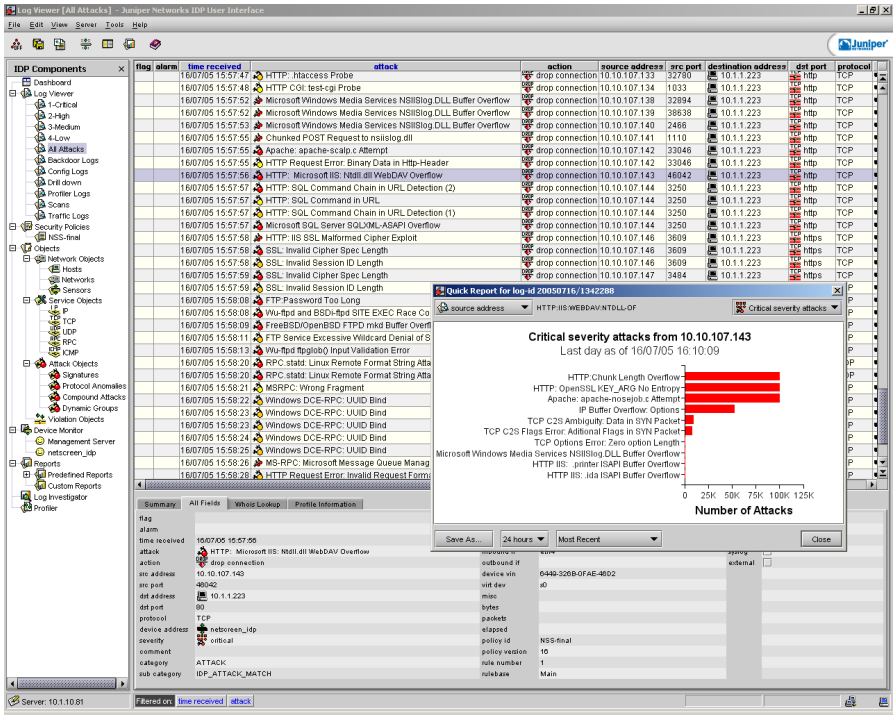

Figure 12 - IDP 600F: Quick Reports from the Log Viewer

There is no way to correlate or group related alerts together, however, either automatically or manually.

It is possible to create multiple unique, customised views of the log data within the Log Viewer by:

- *Changing the column settings (reorder columns)*
- *Filtering by column headings (hide columns)*
- *Filtering by cell content (display all records with a particular source IP address, or ignore all records targeted at a particular host)*
- *Collapsing and/or hiding similar log records to aid readability*

Any number of such customised views can be saved along with all applicable filters enabling them to be recalled and used repeatedly from the Log Viewer menu.

It is also possible to export log records for use in other applications - PDF and Postscript options are available from the GUI, and export to XML, CSV, SNMP, SMTP, Syslog, script, or PostgreSQL database format can be achieved via the command line interface.

The latest drill-down operation is always saved in the *Drill Down* window in the Log Viewer menu hierarchy, thus providing a means for the administrator to switch quickly between drill-down data and higher-level information. Multiple instances of the Log Viewer can also be spawned, each in a new window and each containing unique views of the log data.

In addition to alert entries, the Log Viewer also provides access to *Profiler Logs* (information on individual hosts acquired by the Enterprise System Profiler (ESP)), *Scans*, and Traffic Logs (including fragmentation errors, flow-related errors, and so on).

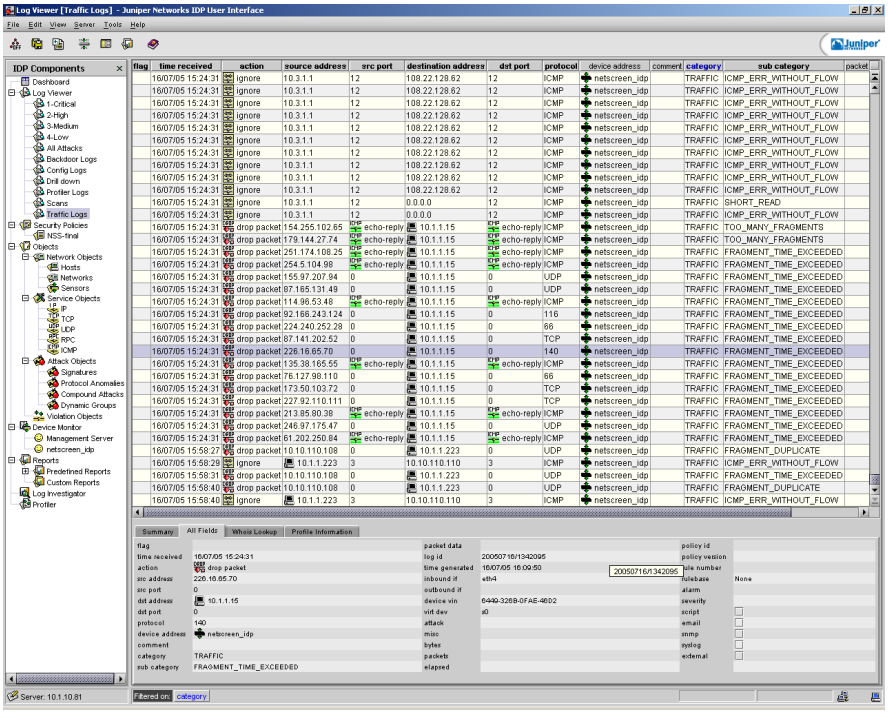

Figure 13 - IDP 600F: Viewing Traffic Logs

New log files are created each day automatically. Log records which are no longer needed can be purged from the Management Server database. The administrator can also set the IDP system to automatically purge logs when the Management Server disk space becomes low. Log entries are lost once purged - there is no archive option.

Reporting and Analysis

There are a number of pre-defined graphical reports included in the UI which are based on log records that summarise threats and traffic behaviour over time.

These include the following:

- *Top Attacks (all/prevented over 24 hours)*
- *Top Attackers (all/prevented over 24 hours)*
- *Top Targets (all/prevented over 24 hours)*
- *Attacks by Severity (all/prevented over 24 hours)*
- *Attacks Over Time (all prevented over 7/30 days)*
- *Critical Attacks (all/prevented over 24 hours)*
- *Critical Thru Medium Attacks (all/prevented over 24 hours)*
- *Top Scan Sources (last 7 days)*
- *Top Scan Targets (last 7 days)*
- *Profiler - New Hosts (last 7 days)*
- *Profiler - New Ports (last 7 days)*
- *Profiler - New Protocols (last 7 days)*
- *Top Rules*
- *Logs by User-set Flag*

Reports present log record data using Log Viewer filters. After selecting a report it is possible to set individual options for its appearance, including selecting a date range and chart style. It is also possible to view the log records on which the report table entries are based (by right-clicking on the data to view and selecting the "*View in Log Viewer*" option), allowing the administrator to "drill down" quickly from a high level graphical summary to a detailed view.

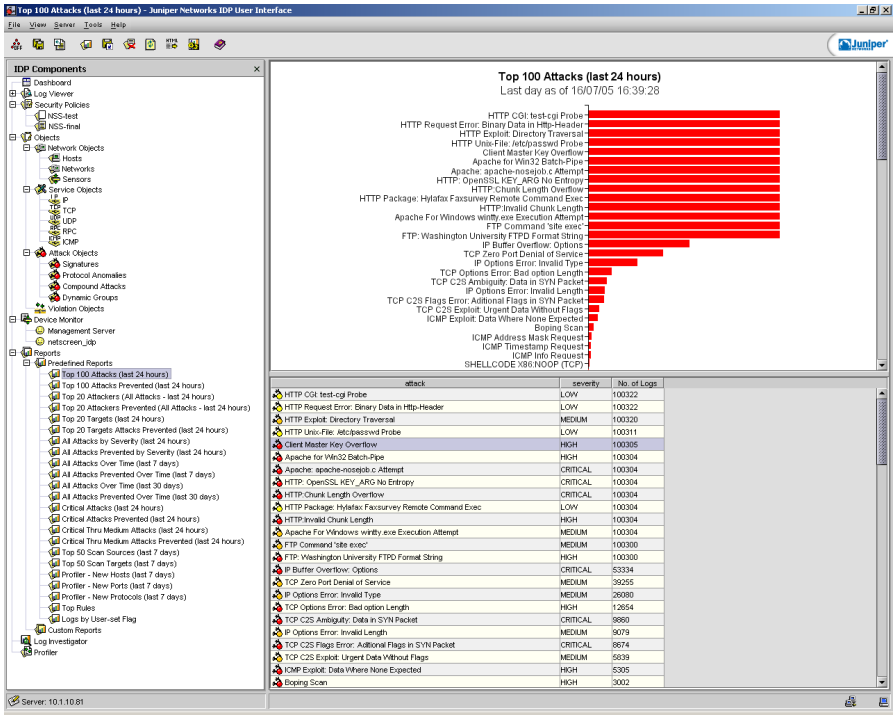

Figure 14 - IDP 600F: Running reports

Custom reports provide the administrator with the means to specify his own selection criteria, report content (from a list of data collected for each alert), time period and report style (count or time based, bar or pie chart).

These can be named and saved in the *Custom Report* section of the main menu tree. In addition, "*Quick Reports*" can be created rapidly from the *Alert Viewer* based on criteria taken directly from the alert being viewed at the time, and these too can be saved in the *Custom Reports* section for later use.

The *Log Investigator* is designed along the lines of a dynamic “reporting spreadsheet” and is intended to provide the administrator with the means to “slice and dice” log entries in various ways. It allows him to select different data sets via the use of filters and include/exclude options to incrementally refine the search for a specific range of entries.

The Log Investigator uses a default filter that selects only log records with the value of ATTACK in the category column of the Log Viewer. The administrator can choose which criteria he wishes to see on the left and right axes of the table, and then set additional filters to narrow the focus of his analysis of the data. For example, he might choose to filter by the *flag* column and have the Log Investigator only consider log records with *ATTACK* in the category column and a *HIGH* setting in the flag column.

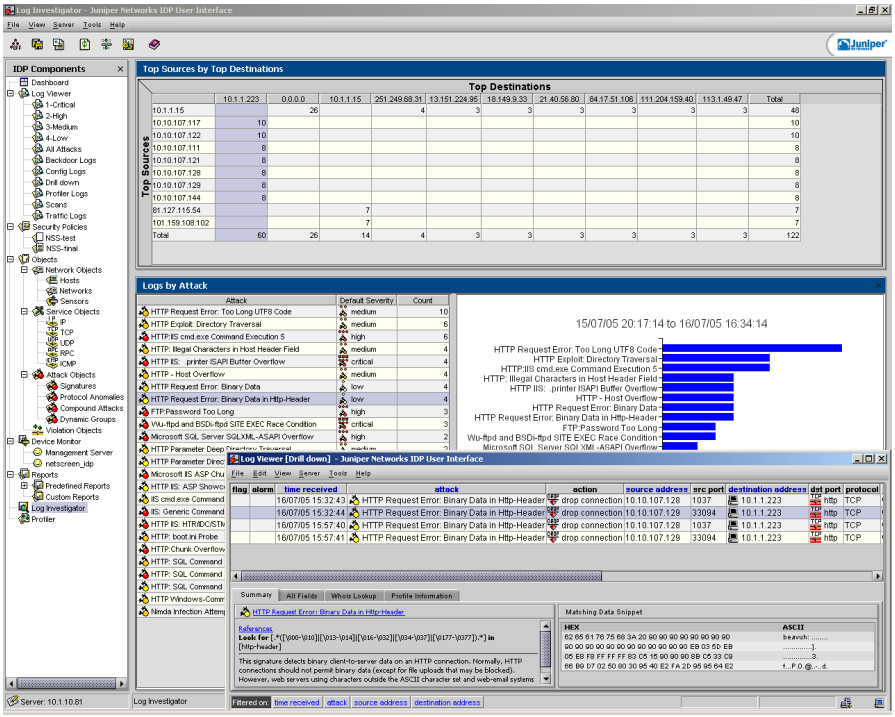

Figure 15 - IDP 600F: Log Investigator

For individual table cells, or entire rows and columns, the administrator can “zoom in” to focus on the destination ports, attacks, or time. The Log Viewer then appears in a separate window, displaying the log records which produced the entries in the Log Investigator. The filters that produced the Log Investigator entries appear on the status bar.

The *Enterprise Security Profiler* (ESP) is intended to help administrators monitor their network security posture by collecting forensic data on network traffic and storing that data to form a *network profile*. Data can then be used to identify changes in network behaviour, which in turn can be translated into policy modifications. In addition to being a proactive analysis and planning tool, data collected by ESP provides administrators with a useful post attack analysis tool.

The way ESP works is that it stores data already collected during the attack detection process, and makes that available via a number of database views and filters. The administrator needs to first configure the system to collect Profiler data - it does not do this by default (since this would have an impact on overall performance of the sensor - note that the Profiler was disabled for our performance testing).

ESP collects data at both the network layer and the application layer. Network information collected includes items such as IP addresses (host connected to and from), and port number, whilst application level data includes items such as the actual application used over the network (IE, Apache, IIS, and so on), version number, user name, and URL.

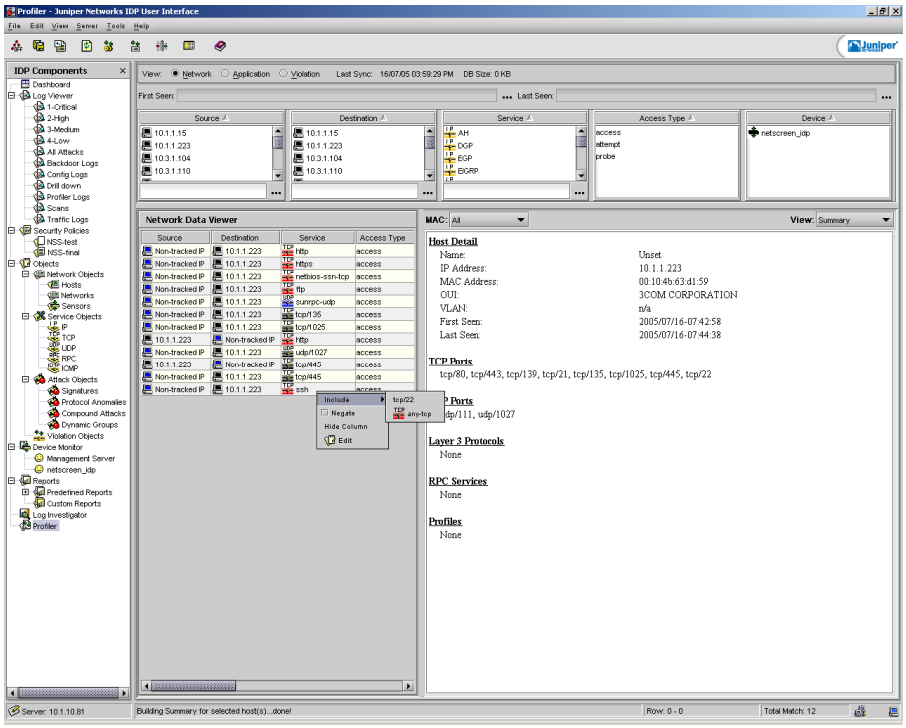

Figure 16 - IDP 600F: Enterprise Security Profiler (ESP)

Data on individual hosts (IP address, MAC address, host name, protocols used, ports used, etc.) is stored and displayed within the *Profiler* (it has its own menu option) as well as against each alert in the Alert Viewer, where it populates its own *Profile Information* tab in the alert details window at the bottom of the main Alert Viewer screen.

Once the Profile Information tab has been selected for a particular alert, the administrator can choose from a number of display options in a drop down menu, offering host detail, TCP ports used, UDP ports used, RPC services, layer 3 protocols used, and details of other machines with which the host has communicated. The *Profiler* view - selectable via the main menu tree - provides a direct view into the Profiler database.

A number of selection criteria are available to enable the administrator to examine which clients are communicating with which servers, which protocols are being used, and which applications are being run over his network.

There are three different views in the Profiler to help administrators zero in on the information they need: the *Network View* for layer 3 and 4 information, *Application View* for layer 7 data, and *Violation View* for pseudo firewall policies that help pinpoint what an administrator may not know about the network. The Network View provides data filtered and displayed on source and destination IP address, port number, MAC address, and so on.

The Application View can often be much more interesting, since it displays the context of each suspicious activity, including the server name and version, the actual command issued, the URL requested, and so on. Thus it is a simple matter to focus on FTP servers, then select all those occasions where someone tried to log in as *root* or *administrator*. Likewise, it would be a simple matter to focus on a particular version of Apache or IIS, and then see what activity has been aimed at that particular server or group of servers.

This information can then be used to set *Violation* rules, to ensure that only valid client-server communications are allowed across the network. For example, the administrator can define that FTP traffic is allowed only between one particular group of IP addresses and one particular FTP server, or that Telnet traffic is not allowed at all. This way, IDP can detect and prevent malicious traffic based purely on policy violations rather than specific exploits. The *Enterprise Security Profiler* is an extremely useful feature, and one that goes beyond pure *intrusion detection* or *prevention*, and into the realms of *policy enforcement*.

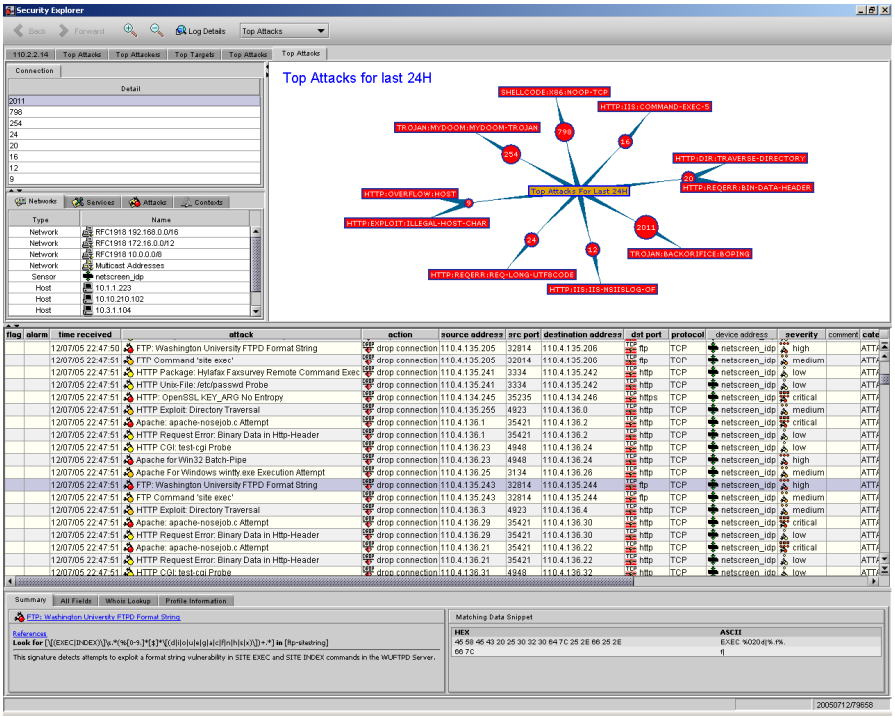

Figure 17 - IDP 600F: Security Explorer

The final forensic and reporting tool is a recent addition to the product - the *Security Explorer*.

By selecting a particular IP address from the source or destination field of any alert using *ALT-left mouse click*, the administrator gains immediate access to a number of different views of all the data pertaining to that address.

The most striking is the useful graphical representation of the relationships between the selected host and all other hosts with which it has interacted or attacks which have involved it. The view is selected from a drop-down menu which allows the administrator to select *Top Attacks, Top Attackers* or *Top Targets* in the last 24 hours.

The graphical display “swims” around the pane, reorienting itself depending on which node (or connection between two nodes) is selected as the main point of focus, and any node can be double-clicked to bring up a new tab with the focus on the node selected. Selecting various points on the graphical view brings up summary information of the active services on a particular host, network addresses used, attacks seen and context information. Below this is the normal Log Viewer display containing a list of all the alerts pertaining to the IP address selected.

This free-format display is actually more difficult to explain than it is to use, but it can be an extremely powerful alert handling and forensic tool, allowing the administrator to simply “wander around” the most recent data pertaining to a particular host at will.

Verdict

Performance

The IDP 600F is rated at 500Mbps when deployed in-line, and performance at almost all levels of our tests was very good, with 100 per cent of all attacks being detected and blocked under all but the most extreme (10,000 connections per second) load conditions.

Even when pushed beyond its limits (8,000cps), the IDP 600F continued to block all malicious traffic successfully. We would happily confirm Juniper Network’s 500Mbps rating for this device under normal network conditions, and would deem it conservative in a typical network environment.

Basic latency figures were very good for a device of this type at all traffic loads and packet sizes. Behaviour through all of the tests with no background traffic and with a half load of 250Mbps of HTTP traffic was very predictable, and HTTP response times were also very good.

Overall, the results were excellent for a 500Mbps device, and we believe that the IDP 600F could be deployed anywhere on a 500Mbps network, either internally or at the perimeter.

Whether or not the SYN Protector was enabled, the IDP 600F was unable to handle the 50Mbps of SYN flood traffic we launched during our DOS/DDOS tests, and until a more effective SYN protection mechanism is implemented, we recommend the IDP 600F is deployed behind an attack mitigation device or firewall with full DOS/DDOS mitigation capabilities.

The IDP 600F performed consistently and completely reliably throughout our tests, and resistance to ISIC-generated traffic was complete.

Security Effectiveness

Signature recognition (with blocking disabled) was excellent out of the box at 95 per cent. This figure was increased to 97 per cent following a signature pack update which Juniper provided within 48 hours. Blocking performance was identical throughout the tests. Note that with the HTTP server-to-client signatures enabled, the attack recognition rose to a perfect 100 per cent (though with a corresponding decrease in maximum performance as a result).

It is unfortunate that the HTTP server-to-client signatures come with such a performance impact, but at least Juniper includes this capability, giving the administrator the choice between total coverage at the expense of performance, or slightly reduced coverage (irrelevant in a purely internal deployment) with maximum performance.

All of our “false negative” (modified exploit) cases were detected correctly out of the box, demonstrating that the Juniper signatures are generally designed to detect the underlying vulnerability rather than a specific exploit.

The IDP 600F blocked three of our false positive test cases out of the box. However, this was rectified following the signature update, and we noted no further false positives throughout the remainder of the testing.

Resistance to known evasion techniques was excellent, with the IDP 600F achieving a clean sweep across the board in all our main evasion tests. *Fragroute, Whisker, ADMmutate* and even *RPC record fragging* all failed to trick the IDP 600F into ignoring valid attacks.

Out of the box, the IDP-600F handled 100,000 open connections, and 500,000 connections after tuning (the supported maximum is 220,000) - acceptable for a 500Mbps device. Default operation of the device is to block new connections when the state tables are full or resources are low, and this behaviour is not configurable. It would be nice to see a choice offered to the administrator of whether to age out old connections or block new ones.

The IDP 600F comes with a number of suggested Protection Policy templates to aid the administrator when setting up initially - these include suggested policies for file servers, Web servers, DNS servers, and so on, along with a typical “default” policy.

These are based on a number of carefully constructed Dynamic Groups which will allow Juniper to modify these policies with each signature update, adding new signatures automatically, or moving signatures between *alert* and *block* modes as confidence in a new signature increases.

Usability

In terms of day to day operation, the IDP 600F demonstrates high levels of usability with an excellent set of centralised management tools. The three tier architecture should scale well, particularly given the claims for the proprietary inter-device communications protocol and high-performance alert logging database. We did notice that this management solution performed significantly better than others in our labs when under heavy attack, with little or no interruption to communication between Sensor and Management Server.

The management interface is very straightforward to use, allowing the administrator to create and modify Security Policies easily. Signature updates occur daily and the update procedure is excellent, allowing the administrator to “vet” the changes before new signatures are downloaded. The facility to create custom signatures is available, and is also very straightforward to use.

An extensive search facility within the Policy Editor provides the administrator with the means to effect very precise searches for specific signatures, which can then be easily copied or modified. In this respect, Juniper Networks IDP is one of the most open systems we have seen, and the *Compound Signature* capability provides the means to create some very complex and accurate signatures.

Once Policies have been created or updated, a single button provides the means to deploy them to all Sensors in a single operation. One really nice feature is the automatic version control, with a complete audit trail of all previous Policies applied to each Sensor, and the ability to roll back to a previous version if required.

The administrator is provided with a high degree of flexibility and control in how Policies are defined, stored and deployed (with the ability to configure individual rules within a Policy for individual Sensors), and we found Policy management to be very strong in this product.

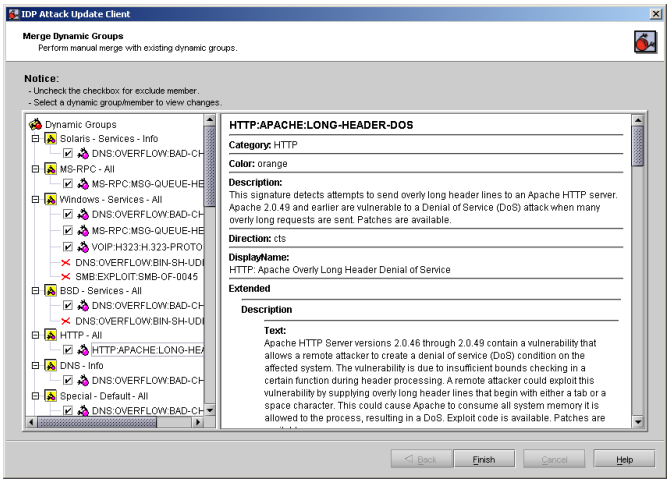

Figure 18 - IDP 600F: Signature update process

Alert handling, too, was well implemented. Alerts appear quickly at the UI Console, and the descriptions are generally very accurate. The ability to apply filters quickly and easily (often by simply right-clicking on a value in an alert and selecting an option from a drop-down menu) and drill down to more detailed views within the Log Viewer and Log Investigator makes it a useful tool for the forensic investigator.

A wealth of useful forensic/reporting features include the ability to create custom views and reports, *Quick Reports* (created from the Alert Viewer), the unusual *Security Explorer*, and the excellent *Enterprise Security Profiler*. The latter provides the means for the administrator to take IDP beyond pure *intrusion detection* and *prevention*, and into the realms of *policy enforcement*, increasing its usefulness in many corporate environments.

All of these combine to make the IDP 600F one of the strongest products in terms of alert handling, reporting and forensic analysis that we have seen.

Contact Details

Company: Juniper Networks, Inc.

E-mail: http://www.juniper.net/howtobuy/

Internet: http://www.juniper.net

Address:
1194 N. Mathilda Ave, Building 3
Sunnyvale
CA 94089
USA

Tel: 866 298 6428 (inside USA), or +1 978 589 0500 (outside USA).

Fax: +1 408 745 2100

APPENDIX A – TEST RESULTS

The aim of this procedure is to provide a thorough test of all the main components of an in-line Intrusion Prevention System (IPS) device in a controlled and repeatable manner and in the most “real world” environment that can be simulated in a test lab.

The Test Environment

The network is 100/1000Mbit Ethernet with CAT 5e cabling and Cisco 6500-Series switches (these have a mix of fibre and copper Gigabit interfaces). All devices are expected to be provided as appliances - if software-only, the supplier pre-installs the software on the recommended hardware platform. The sensor is configured as a perimeter device during testing (i.e. as if installed behind the main Internet gateway/firewall). There is no firewall protecting the target subnet.

Traffic generation equipment - such as the machines generating exploits, Spirent Avalanche and Spirent Smartbits *transmit* port - is connected to the “external” network, whilst the “receiving” equipment - such as the “target” hosts for the exploits, Spirent Reflector and Spirent Smartbits *receive* port - is connected to the internal network. The device under test is connected between two “gateway” switches - one at the edge of the external network, and one at the edge of the internal network.

All “normal” network traffic, background load traffic and exploit traffic will therefore be transmitted **through** the device under test, from external to internal. The same traffic is mirrored to a single SPAN port of the external gateway switch, to which an Adtech network monitoring device is connected. The Adtech AX/4000 monitors the same mirrored traffic to ensure that the total amount of traffic never exceeds 1Gbps (which would invalidate the test run).

The management interface is used to connect the appliance to the management console on a private subnet. This ensures that the sensor and console can communicate even when the target subnet is subjected to heavy loads, in addition to preventing attacks on the console itself.

Section 1 – Detection Engine

The aim of this section is to verify that the sensor is capable of detecting and blocking a wide range of common exploits accurately, whilst remaining resistant to false positives. All tests in this section are completed with **no background network load**. The latest signature pack is acquired from the vendor, and sensors are deployed with **all** available attack signatures enabled (some audit/informational signatures may be disabled).

Test 1.1 - Attack Recognition

Whilst it is not possible to validate completely the entire signature set of any sensor, this test attempts to demonstrate how accurately the sensor detects and blocks a wide range of common exploits, port scans, and Denial of Service attempts. These are updated/changed for every new test, and all exploits are run with no load on the network and no IP fragmentation.

Our attack suite contains over 100 basic exploits (plus variants) covering the following areas:

- ***Test 1.1.1 -*** *Backdoors (standard ports and random ports)*
- ***Test 1.1.2 -*** *DNS/WINS*
- ***Test 1.1.3 -*** *DOS*
- ***Test 1.1.4 -*** *False negatives (common exploits which have been modified to remove or alter obvious "triggers" - this ensures that the signatures are coded for the underlying vulnerability rather than a particular exploit)*
- ***Test 1.1.5 -*** *Finger*
- ***Test 1.1.6 -*** *FTP*
- ***Test 1.1.7 -*** *HTTP*
- ***Test 1.1.8 -*** *ICMP (including unsolicited ICMP response)*
- ***Test 1.1.9 -*** *Reconnaissance*
- ***Test 1.1.10 -*** *RPC*
- ***Test 1.1.11 -*** *SSH*
- ***Test 1.1.12 -*** *Telnet*
- ***Test 1.1.13 -*** *Database*
- ***Test 1.1.14 -*** *Mail*
- ***Test 1.1.15 -*** *Voice*

A wide range of vulnerable target operating systems and applications are used, and the majority of the attacks are successful, gaining root shell or administrator privileges on the target machine.

We expect all the attacks to be reported in as straightforward and clear a manner as possible (i.e. an "RDS MDAC attack" should be reported as such, rather than a "Generic IIS Attack"). Wherever possible, attacks should be identified by their assigned CVE reference. It will also be noted when a response to an exploit is considered too "noisy", generating multiple similar or identical alerts for the same attack. Finally, we will note whether the device blocks the attack packet only or the entire "suspicious" TCP session.

This test is repeated twice: the first run with blocking disabled on the sensor (monitor mode only) in order to determine which attacks are detected and how accurately they are detected (*Attack Recognition Rating*); the second run with blocking enabled in order to determine which attacks are blocked successfully regardless of how they are detected or what alerts are raised (*Attack Blocking Rating*)

The "**default**" *Attack Recognition Rating-Detect Only* (ARRD) and *Attack Recognition Rating-Block* (ARRB) are each expressed as a percentage of detected/blocked exploits against total number of exploits launched with the default signature set as received by NSS. This demonstrates how effective the sensor can be when simply deploying the default configuration.

Following the initial test run, each vendor is provided with a list of CVE references of the attacks missed, and is then allowed 48 hours to produce an updated signature set. This updated signature set **must** be released to the general public as a standard signature/product update before the report is published - this ensures that vendors do not attempt to code signatures just for this test.

The sensor is then exposed to a second round of identical tests and the "**custom**" ARRD/ARRB is determined. This demonstrates how effective the vendor is at responding to a requirement for new or updated signatures.

Both the *default* and *custom* ARRD/ARRB figures are reported.

Test 1.2 - Resistance To False Positives

The aim of this test is to demonstrate how likely it is that a sensor raises a false positive alert - particularly critical for IPS devices.

We have a number of trace files of normal traffic with "suspicious" content, together with several "neutered" exploits which have been rendered completely ineffective. If a signature has been coded for a specific piece of exploit code rather then the underlying vulnerability, or if it relies purely on pattern matching, some of these false alarms could be alerted upon.

The product attains a "PASS" for each test case if it does **not** raise an alert and does **not** block the traffic. Raising an alert on any of these test cases is considered a "FAIL", since none of the "exploits" used in this test represents a genuine threat. A "FAIL" would thus indicate the chance that the sensor could block legitimate traffic inadvertently.

- ***Test 1.2.1 -*** *False positives*

Section 2 – Evasion

The aim of this section is to verify that the sensor is capable of detecting and blocking basic exploits when subjected to varying common evasion techniques.

Test 2.1 - Baselines

The aim of this test is to establish that the sensor is capable of detecting and blocking a number of common basic attacks (our baseline suite) in their normal state, with no evasion techniques applied. Note that common/older attacks have been chosen deliberately for this particular test to ensure that ALL products tested have signatures in place for the evasion tests.

- ***Test 2.1.1 -*** *Baseline attack replay*

Test 2.2 - Packet Fragmentation and Stream Segmentation

The baseline HTTP attacks are repeated, running them through fragroute using various evasion techniques, including:

- ***Test 2.2.1 -*** *IP fragmentation - ordered 8 byte fragments*
- ***Test 2.2.2 -*** *IP fragmentation - ordered 24 byte fragments*
- ***Test 2.2.3 -*** *IP fragmentation - out of order 8 byte fragments*
- ***Test 2.2.4 -*** *IP fragmentation - ordered 8 byte fragments, duplicate last packet*
- ***Test 2.2.5 -*** *IP fragmentation - out of order 8 byte fragments, duplicate last packet*
- ***Test 2.2.6 -*** *IP fragmentation - ordered 8 byte fragments, reorder fragments in reverse*

- ***Test 2.2.7 -*** *IP fragmentation - ordered 16 byte fragments, fragment overlap (favour new)*
- ***Test 2.2.8 -*** *IP fragmentation - ordered 16 byte fragments, fragment overlap (favour old)*
- ***Test 2.2.9 -*** *TCP segmentation - ordered 1 byte segments, interleaved duplicate segments with invalid TCP checksums*
- ***Test 2.2.10 -*** *TCP segmentation - ordered 1 byte segments, interleaved duplicate segments with null TCP control flags*
- ***Test 2.2.11 -*** *TCP segmentation - ordered 1 byte segments, interleaved duplicate segments with requests to resync sequence numbers mid-stream*
- ***Test 2.2.12 -*** *TCP segmentation - ordered 1 byte segments, duplicate last packet*
- ***Test 2.2.13 -*** *TCP segmentation - ordered 2 byte segments, segment overlap (favour new)*
- ***Test 2.2.14 -*** *TCP segmentation - ordered 1 byte segments, interleaved duplicate segments with out-of-window sequence numbers*
- ***Test 2.2.15 -*** *TCP segmentation - out of order 1 byte segments*
- ***Test 2.2.16 -*** *TCP segmentation - out of order 1 byte segments, interleaved duplicate segments with faked retransmits*
- ***Test 2.2.17 -*** *TCP segmentation - ordered 1 byte segments, segment overlap (favour new)*
- ***Test 2.2.18 -*** *TCP segmentation - out of order 1 byte segments, PAWS elimination (interleaved dup segs with older TCP timestamp options)*
- ***Test 2.2.19 -*** *IP fragmentation - out of order 8 byte fragments, interleaved duplicate packets scheduled for later delivery*
- ***Test 2.2.20 -*** *TCP segmentation - ordered 16 byte segments, segment overlap (favour new (Unix))*

For each of the evasion techniques, we note if (i) the attempted attack is blocked successfully (the primary aim of any IPS device), (ii) the attempted attack is detected and an alert raised in **any** form, and (iii) if the exploit is successfully “decoded” to provide an accurate alert relating to the original exploit, rather than alerting purely on anomalous traffic detected as a result of the evasion technique itself.

Test 2.3 - URL Obfuscation

The baseline HTTP attacks are repeated, this time applying various URL obfuscation techniques made popular by the Whisker Web server vulnerability scanner, including:

- ***Test 2.3.1 -*** *URL encoding*
- ***Test 2.3.2 -*** */./ directory insertion*
- ***Test 2.3.3 -*** *Premature URL ending*
- ***Test 2.3.4 -*** *Long URL*
- ***Test 2.3.5 -*** *Fake parameter*
- ***Test 2.3.6 -*** *TAB separation*
- ***Test 2.3.7 -*** *Case sensitivity*
- ***Test 2.3.8 -*** *Windows \ delimiter*
- ***Test 2.3.9 -*** *Session splicing*

For each of the evasion techniques, we note if (i) the attempted attack is blocked successfully, (ii) the attempted attack is detected and an alert raised in **any** form, and (iii) if the exploit is successfully "decoded" to provide an accurate alert relating to the original exploit, rather than alerting purely on anomalous traffic detected as a result of the evasion technique itself.

Test 2.4 - Miscellaneous Evasion Techniques

Certain baseline attacks are repeated, and are subjected to various protocol- or exploit-specific evasion techniques, including:

- ***Test 2.4.1 -*** *Altering default ports/passwords for backdoors*
- ***Test 2.4.2 -*** *Inserting spaces in FTP command lines*
- ***Test 2.4.3 -*** *Inserting non-text Telnet opcodes in FTP data stream*
- ***Test 2.4.4 -*** *Polymorphic mutation (ADMmutate)*
- ***Test 2.4.5 -*** *Altering protocol and RPC PROC numbers*
- ***Test 2.4.6 -*** *RPC record fragging (MS-RPC and Sun)*
- ***Test 2.4.7 -*** *HTTP exploits to non-standard port*

For each of the evasion techniques, we note if (i) the attempted attack is blocked successfully, (ii) the attempted attack is detected and an alert raised in **any** form, and (iii) if the exploit is successfully "decoded" to provide an accurate alert relating to the original exploit, rather than alerting purely on anomalous traffic detected as a result of the evasion technique itself.

Section 3 – Stateful Operation

The aim of this section is to be able to determine whether the sensor is capable of monitoring stateful sessions established through the device at various traffic loads without either losing state or incorrectly inferring state.

Test 3.1 - Stateless Attack Replay (Mid-Flows)

This test determines whether the sensor is resistant to stateless attack flooding tools - these utilities are used to generate large numbers of false alerts on the protected subnet using valid source and destination addresses and a range of protocols.

The main characteristic of many flooding tools is the fact that they generate single packets containing "trigger" patterns without first attempting to establish a connection with the target server. Whilst this can be effective in raising alerts with some stateless protocols such as UDP and ICMP, they should never be capable of raising an alert for exploits based on stateful protocols such as FTP and HTTP.

In this test, we transmit a number of packets taken from capture files of valid exploits, but without first establishing a valid session with the target server. We also remove the session tear down and acknowledgement packets so that the sensor can not "infer" that a valid connection was made.

In order to receive a "PASS" in this test, no alerts should be raised for any of the actual exploits (although "mid-flow" alerts are permitted).

However, each packet should be blocked if possible since it represents a "broken" or "incomplete" session.

- ***Test 3.1.1 -*** *Stateless attack replay*

Test 3.2 - Simultaneous Open Connections (default settings)

This test determines whether the sensor is capable of preserving state across increasing numbers of open connections, as well as continuing to detect and block new exploits when the state tables are filled. It also attempts to determine whether or not the sensor will block legitimate traffic once state tables are filled. This test is run using the default sensor settings (no tuning of sensor parameters).

A legitimate HTTP session is opened and the first packet of a two-packet exploit is transmitted. The Spirent Avalanche (on the "external" interface of the sensor) then opens various numbers of TCP sessions from 10,000 to 1,000,000 (one million) with the Spirent Reflector (on the "internal" interface of the sensor). The initial HTTP session is then completed with the second half of the exploit and the session is closed. If the sensor is still maintaining state on the first session established, the exploit will be recorded. If the state tables have been exhausted, the exploit string will be seen as a non-stateful attack, and will thus be ignored.

Both halves of the exploit are required to trigger an alert - a product will fail the test if it fails to generate an alert after the second packet is transmitted, or if it raises an alert on either half of the exploit on its own.

At each step, we ensure that the sensor is still capable of detecting and blocking freshly-launched exploits once all the connections are open, as well as confirming that the device does not block legitimate traffic (perhaps as a result of state tables filling up). We then launch further exploits whilst the Avalanche/Reflector devices "churn" connections at the maximum level set, ensuring that the sensor is still capable of detecting and blocking freshly-launched exploits as old connections are torn down and new ones recreated constantly.

- ***Test 3.2.1 - Attack Detection:*** *This test ensures that the sensor continues to detect new exploits as the number of open sessions is increased in stages from 10,000 to 1,000,000*
- ***Test 3.2.2 - Attack Blocking:*** *This test ensures that the sensor continues to block new exploits as the number of open sessions is increased in stages from 10,000 to 1,000,000*
- ***Test 3.2.3 - State Preservation:*** *This test ensures that the sensor maintains the state of pre-existing sessions as the number of open sessions is increased in stages from 10,000 to 1,000,000*
- ***Test 3.2.4 - Legitimate Traffic Blocking:*** *This test ensures that the sensor does not begin to block legitimate traffic as the number of open sessions is increased in stages from 10,000 to 1,000,000*

Test 3.3 - Simultaneous Open Connections (after tuning)

Test 3.2 is repeated after any tuning recommended by the vendor (if applicable) to increase the size of the state tables.

- ***Test 3.3.1 - Attack Detection:*** *As Test 3.2.1 following tuning*
- ***Test 3.3.2 - Attack Blocking:*** *As Test 3.2.2 following tuning*
- ***Test 3.3.3 - State Preservation:*** *As Test 3.2.3 following tuning*
- ***Test 3.3.4 - Legitimate Traffic Blocking:*** *As Test 3.2.4 following tuning*

Section 4 – Detection/Blocking Performance Under Load

The aim of this section is to verify that the sensor is capable of detecting and blocking exploits when subjected to increasing loads of background traffic up to the maximum bandwidth supported as claimed by the vendor.

The latest signature pack is acquired from the vendor, and sensors are deployed with **all** available attack signatures enabled (some audit/informational signatures may be disabled). Each sensor is configured to **detect and block** suspicious traffic.

Our “attacker” host launches a fixed number of exploits at a target host on the subnet being protected by the device under test. The Adtech network monitor is configured to monitor the switch SPAN port consisting of normal, exploit and background traffic, and is capable of reporting the total number of exploit packets seen on the wire as verification.

A fixed number of exploits are launched with zero background traffic to ensure the sensor is capable of detecting our baseline attacks. Once that has been established, increasing levels of varying types of background traffic are generated **through** the sensor in order to determine the point at which the sensor begins to miss attacks - all tests are repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic (or up to the maximum rated throughput of the device should this be less than 1Gbps).

At all stages, the Adtech network monitor verifies both the overall traffic loading and the total number of exploits seen on the target subnet. An additional confirmation is provided by the target host which reports the number of exploits which actually made it through.

The *Attack Blocking Rate* (ABR) at each background load is expressed as a percentage of the number of exploits blocked by the sensor (when in blocking mode) against the number verified by the Adtech network monitor and target host. The *Attack Detection Rate* (ADR) at each background load is expressed as a percentage of the number of exploits detected by the sensor (with blocking mode disabled) against the number verified by the Adtech network monitor and target host.

For each type of background traffic, we also determine the maximum load the sensor can sustain before it begins to drop packets/miss alerts. It is worth noting that devices which demonstrate 100 per cent ABR (blocking) but less than 100 per cent ADR (detection) in these tests will be prone to blocking **legitimate** traffic under similar loads.

Test 4.1 - UDP Traffic To Random Valid Ports

This test uses UDP packets of varying sizes generated by a **Smartbits SMB6000** with LAN-3301A 10/100/1000Mbps **TeraMetrics** cards installed.

A constant stream of the appropriate mix of packets - with variable source IP addresses and ports transmitting to a single fixed IP address/port - is transmitted through the sensor (bi-directionally, maximum of 1Gbps).

Each packet contains dummy data, and is targeted at a valid port on a valid IP address on the target subnet. The percentage load and packets per second (pps) figures are verified by the Adtech Gigabit network monitoring tool before each test begins. Multiple tests are run and averages taken where necessary.

This traffic does not attempt to simulate any form of “real world” network condition. The aim of this test is purely to determine the raw packet processing capability of the sensor, and its effectiveness at passing “useless” packets quickly in order to pass potential attack packets to the detection engine. The range of packet sizes has been selected to mirror the maximum, minimum and average packet sizes used in our HTTP stress tests.

- ***Test 4.1.1 - 256 byte packets - maximum 453,000 packets per second:*** *This test is roughly equivalent to a 40,000 connections per second test in our HTTP stress tests (in terms of packet size and packets per second rate), and has been included to provide an indication of the packet processing performance under the most extreme conditions for most devices - it is unlikely that any real-life network will ever see network loads of over 450,000 256-byte packets per second unless under severe DOS conditions. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic.*
- ***Test 4.1.2 - 550 byte packets - maximum 220,000 packets per second:*** *This test has been included to provide a comparison with our “real world” packet mixes, since the average packet size is similar. No sessions are created during this test and there is very little for the detection engine to do in the way of protocol analysis. This test provides a reasonable indication of the ability of a device to process packets from the wire on an “average” network, and we would expect all products to demonstrate good performance levels. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic.*
- ***Test 4.1.3 - 1000 byte packets - maximum 122,000 packets per second:*** *This test is the complete opposite of the 256 byte packet test, in that we would expect every single product to be capable of returning 100 per cent detection rates across the board when using only 1000 byte packets. We have included this test mainly to demonstrate how easy it is to achieve good results using large packets – beware of test results that* ***only*** *quote performance figures using similar (or larger) packet sizes. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic.*

Test 4.2 - HTTP “Maximum Stress” Traffic With No Transaction Delays

HTTP is the most widely used protocol in most normal networks, as well as being one of the most widely exploited. The number of potential HTTP exploits for the protocol makes a pure HTTP network something of a torture test for the average sensor.

The use of multiple Spirent Communications **Avalanche 2500** and **Reflector 2500** devices allows us to create true “real world” traffic at speeds of up to 4.2 Gbps as a background load for our tests. Our Avalanche configuration is capable of simulating over 5 million users, with over 5 million concurrent sessions, and over 200,000 HTTP requests per second.

By creating genuine session-based traffic with varying session lengths, the sensor is forced to track valid sessions, thus ensuring a higher workload than for simple packet-based background traffic. This provides a test environment that is as close to “real world” as it is possible to achieve in a lab environment, whilst ensuring absolute accuracy and repeatability.

The aim of this test is to stress the HTTP detection engine and determine how the sensor copes with detecting and blocking exploits under network loads of varying average packet size and varying connections per second.

Each transaction consists of a single HTTP GET request and there are no transaction delays (i.e. the Web server responds immediately to all requests). All packets contain valid payload (a mix of binary and ASCII objects) and address data, and this test provides an excellent representation of a live network (albeit one biased towards HTTP traffic) at various network loads.

- ***Test 4.2.1 -*** *Max 2,500 new connections per second - average packet size 1000 bytes - maximum 120,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic. With relatively low connection rates and large packet sizes, we expect all sensors to achieve 100% blocking rates throughout this test.*
- ***Test 4.2.2 -*** *Max 5,000 new connections per second - average packet size 540 bytes - maximum 225,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic. With average connection rates average packet sizes, this is a good approximation of a real-world production network, and we expect all sensors to perform well in this test.*
- ***Test 4.2.3 -*** *Max 10,000 new connections per second - average packet size 440 bytes - maximum 275,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic. With average packet sizes coupled with very high connection rates, this is a strenuous test for any sensor, and represents a very heavily used production network.*
- ***Test 4.2.4 -*** *Max 20,000 new connections per second - average packet size 360 bytes - maximum 320,000 packets per second. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic. With small packet sizes and extremely high connection rates this is an extreme test for any sensor. Not many sensors will perform well at all levels of this test.*

Test 4.3 - HTTP “Maximum Stress” Traffic With Transaction Delays

This test is identical to Test 4.2 except that we introduce a 10 second delay in the server response for each transaction. This has the effect of maintaining a high number of open connections throughout the test, thus forcing the sensor to utilise additional resources to track those connections.

- ***Test 4.3.1 -*** *Max 5,000 new connections per second - average packet size 540 bytes - maximum 225,000 packets per second - 10 second transaction delay - maximum 50,000 open connections. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic. With average connection rates average packet sizes, this is a good approximation of a real-world production network, and we expect all sensors to perform well in this test.*
- ***Test 4.3.2 -*** *Max 10,000 new connections per second - average packet size 440 bytes - maximum 275,000 packets per second - 10 second transaction delay - maximum 100,000 open connections. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic. With average packet sizes coupled with very high connection rates, this is a strenuous test for any sensor, and represents a very heavily used production network.*

Test 4.4 - Protocol Mix Traffic

Whereas 4.2 and 4.3 provide a pure HTTP environment with varying connection rates and average packet sizes, the aim of this test is to simulate more of a “real world” environment by introducing additional protocols whilst still maintaining a precisely repeatable and consistent background traffic load (something rarely seen in a real world environment).

The result is a background traffic load that, whilst less stressful than previous tests, is closer to what may be found on a heavily-utilised “normal” production network.

- ***Test 4.4.1 -*** *72% HTTP traffic (540 byte packets) + 20% FTP traffic + 6% UDP traffic (256 byte packets). Max 4000 new connections per second - average packet size 540 bytes - maximum 215,000 packets per second - maximum 750 open connections. Repeated with 250Mbps, 500Mbps, 750Mbps and 1000Mbps of background traffic. With lower connection rates, average packets sizes and a common protocol mix, this is a good approximation of a heavily-used production network, and we expect all sensors to perform well throughout this test.*

Test 4.5 - “Real World” Traffic

This is as close as it is possible to come to a true “real world” environment under lab conditions. For this test we eliminate the Reflector device and substitute an IIS Web server installed on a dual-Xeon server with Gigabit interface and 4GB RAM. This server holds a copy of The NSS Group Web site, and is capable of handling a full 1Gbps of traffic. We then capture a typical client browsing session on the NSS Group Web site, accessing a mixture of menu pages, lengthy text-based reports and multiple graphical images (screen shots) and have Avalanche replay multiple identical sessions from up to **20 new users per second**.

It should be noted that whereas the goal of the previous tests is a very predictable, consistent and repeatable background load that never varies, the nature of this test means that traffic is slightly more “bursty” in nature.

- ***Test 4.5.1 - Pure HTTP Traffic (simulated browsing session on NSS Web site):*** *Max 4700 new connections per second - 20 new users per second - average packet size 560 bytes - maximum 210,000 packets per second.*

*Repeated with 250Mbps, 500Mbps, 750Mbps and 950Mbps of background traffic. With genuine server responses to genuine **browser** sessions consisting **of multiple transactions per** session, this is a typical "real world" background load, albeit pure HTTP. Although the Web server and the network are extremely busy at the higher traffic loads, the "normal" connection rates and packet sizes should enable most sensors to perform well at all load levels in this test.*

- ***Test 4.5.2 - Protocol Mix (72% HTTP traffic (simulated browsing sessions as 4.5.1)) + 20% FTP traffic + 6% UDP traffic (256 byte packets)):** Max 3700 new connections per second - average packet size 560 bytes - maximum 205,000 packets per second - maximum 1,500 open connections.*

  *Repeated with 250Mbps, 500Mbps, 750Mbps and 950Mbps of background traffic. With genuine server responses to genuine browser sessions consisting of multiple **transactions** per session, mixed with FTP and UDP traffic, this is a typical "real world" background load. Although the Web server and the network are extremely busy at the higher traffic loads, the "normal" connection rates and packet sizes should enable most sensors to perform well at all load levels in this test.*

To gauge the effects of varying (smaller) packet sizes, connection rates and transaction delays, the results of tests 4.2 - 4.4 should be examined.

Section 5 – Latency & User Response Times

The aim of this section is to determine the effect the sensor has on the traffic passing through it under various load conditions.

Should a device impose a high degree of latency on the packets passing through it, a network or security administrator would need to think carefully about how many devices could be installed in a single data path before user response times became unacceptable or the combination of devices caused excessive timeouts. We also determine the effect of high levels of normal HTTP traffic and a basic DOS attack on the average latency and user response times.

Test 5.1 - Latency

We use Spirent SmartFlow software and The Smartbits SMB6000 with Gigabit TeraMetrics cards to create multiple traffic flows through the appliance and measure the basic throughput, packet loss, and latency through the sensor. This test - whilst not indicative of real-life network traffic - provides an indication of how much the sensor affects the traffic flow through it. This data is particularly useful for network administrators who need to gauge the effect of any form of in-line device which is likely to be placed at critical points within the corporate network.

SmartFlow runs through several iterations of the test varying the traffic load from 250Mbps to 1Gbps bi-directionally (or up to the maximum rated throughput of the device should this be less than 1Gbps) in steps of 250Mbps. This is repeated for a range of packet sizes (256 bytes, 550 bytes and 1000 bytes) of UDP traffic with variable IP addresses and ports. At each iteration of the test, SmartFlow records the number of packets dropped, together with average and maximum latency.

- ***Test 5.1.1 - Latency With No Background Traffic:*** *SmartFlow traffic is passed across the infrastructure switches and through the device (the latency of the basic infrastructure is known and is constant throughout the tests). The packet loss and average latency are recorded at each packet size and each load level from 250Mbps to 1Gbps (in 250Mbps steps).*
- ***Test 5.1.2 - Latency With Background Traffic Load:*** *The Avalanche and Reflector are configured to generate a fixed amount of background HTTP traffic through the sensor (up to 50 per cent of the maximum rated bandwidth of the device under test - maximum 500Mbps - maximum 2,500 new connections per second - average packet size 540 bytes - maximum 112,500 packets per second).*

  A 250Mbps bi-directional load of SmartFlow traffic at various packet sizes (256 bytes, 540 bytes and 1000 bytes) is then passed across the infrastructure switches and through the device and the packet loss and average latency are recorded.
- ***Test 5.1.3 - Latency When Under Attack:*** *The Spirent WebSuite software is used to generate a fixed load of DOS/DDOS traffic of 10 per cent of the maximum rated bandwidth of the device under test (maximum 100Mbps). A 250Mbps bi-directional load of SmartFlow traffic at various packet sizes (256 bytes, 540 bytes and 1000 bytes) is then passed across the infrastructure switches and through the device and the packet loss and average latency are recorded. The device should be configured to detect/block/mitigate the DOS attack by the most efficient method available.*

Test 5.2 - User Response Times

Avalanche and Reflector devices are used to generate HTTP sessions through the device in order to gauge how any increases in latency will impact the user experience in terms of failed connections and increased Web response times.

- ***Test 5.2.1 - Web Response With No Background Traffic:*** *The Avalanche and Reflector are configured to generate HTTP traffic through the sensor (up to 50 per cent of the maximum rated bandwidth of the device under test - maximum 500Mbps - maximum 2,500 new connections per second - average packet size 540 bytes - maximum 112,500 packets per second).*

  The minimum, maximum and average page response times and number of failed connections are recorded by Avalanche to provide an indication of the expected response times under normal traffic conditions.
- ***Test 5.2.2 - Web Response When Under Attack:*** *The Avalanche and Reflector are configured to generate HTTP traffic through the sensor as for Test 5.2.1. The Spirent WebSuite software is then used to generate DOS/DDOS traffic up to 10 per cent of the maximum rated bandwidth of the device under test (maximum 100Mbps).*

  The minimum, maximum and average page response times and number of failed connections are recorded by Avalanche to provide an indication of the expected response times when the device is under attack.

Section 6 – Stability & Reliability

These tests attempt to verify the stability of the device under test under various extreme conditions. Long term stability is particularly important for an in-line IPS device, where failure can produce network outages.

- ***Test 6.1.1 - Blocking Under Extended Attack:*** *For this test, we expose the external interface of the device to a constant stream of alerts over an extended period of time. The device is configured to block and alert, and thus this test provides an indication the effectiveness of both the blocking and alert handling mechanisms. A continuous stream of exploits mixed with some legitimate sessions is transmitted through the device at a maximum of 100Mbps (max 50,000 packets per second, average packet sizes in the range of 120-350 bytes) for 8 hours with no additional background traffic. This is not intended as a stress test in terms of traffic load - merely a reliability test in terms of consistency of blocking performance.*

  The device is expected to remain operational and stable throughout this test, and to block 100 per cent of recognisable exploits, raising an alert for each. Results are presented as a simple PASS/FAIL. If any recognisable exploits are passed - caused by either the volume of traffic ***or*** *the sensor failing open for any reason - this will result in a FAIL.*

- ***Test 6.1.2 - Passing Legitimate Traffic Under Extended Attack:*** *This test is identical to 6.1.1, where we expose the external interface of the device to a constant stream of alerts over an extended period of time. The device is expected to remain operational and stable throughout this test, and to pass 100 per cent of legitimate traffic. Results are presented as a simple PASS/FAIL. If any legitimate traffic is blocked - caused by either the volume of traffic* ***or*** *the sensor failing closed for any reason - this will result in a FAIL.*

- ***Test 6.1.3 - ISIC/ESIC/TCPSIC/UDPSIC/ICMPSIC:*** *This test attempts to stress the protocol stack of the device under test by exposing it to traffic from the ISIC test tool. The ISIC test tool host is connected directly to the external interface of the sensor, and the ISIC target directly to the internal interface. ISIC traffic is transmitted through the sensor (without passing through any other network equipment) and the effects noted. Traffic load is a maximum of 350Mbps and 60,000 packets per second (average packet size is 690 bytes). Results are presented as a simple PASS/FAIL - the device is expected to remain operational and capable of detecting and blocking exploits throughout the test to attain a PASS.*

Section 7 – Management and Configuration

The aim of this section is to determine the features of the management system, together with the ability of the management port on the device under test to resist attack.

Test 7.1 - Management Port

Clearly the ability to manage the alert data collected by the sensor is a critical part of any IDS/IPS system. For this reason, an attacker could decide that it is more effective to attack the management interface of the device than the detection interface.

Given access to the management network, this interface is often more visible and more easily subverted than the detection interface, and with the management interface disabled, the administrator has no means of knowing his network is under attack.

- ***Test 7.1.1 - Open ports:*** *We will scan the open ports and active services on the management interface and report on known vulnerabilities.*
- ***Test 7.1.2 - ISIC/ESIC/TCPSIC/UDPSIC/ICMPSIC:*** *This test attempts to stress the protocol stack of the management interface of the device under test by exposing it to traffic from the ISIC test tool. The ISIC test tool host is connected directly to the management interface of the IPS sensor, and that interface is also the target. ISIC traffic is transmitted to the management interface of the IPS device (without passing through any other network equipment) and the effects noted.*

  Traffic load is a maximum of 350Mbps and 60,000 packets per second (average packet size is 690 bytes). Results are presented as a simple PASS/FAIL - the device is expected to remain (a) operational and capable of detecting and blocking exploits, and (b) capable of communicating in both directions with the management server/console throughout the test to attain a PASS.

  Test 7.1.3 - *We note whether the ISIC attacks themselves are detected by the sensor even though targeted at the management port.*

Juniper Networks IDP 600F V3.1 Test Results

Section 1 - Detection Engine

Test 1.1 – Attack Recognition	Attacks	Default ARRD	Default ARRB	Custom ARRD	Custom ARRB
Test 1.1.1 - Backdoors	7	7	7	7	7
Test 1.1.2 - WINS/DNS	3	3	3	3	3
Test 1.1.3 - DOS	10	10	10	10	10
Test 1.1.4 - False negatives (modified exploits)	14	14	14	14	14
Test 1.1.5 - Finger	4	4	4	4	4
Test 1.1.6 - FTP	5	5	5	5	5
Test 1.1.7 - HTTP	43	40[1]	40[1]	40[1]	40[1]
Test 1.1.8 - ICMP	2	2	2	2	2
Test 1.1.9 - Reconnaissance	8	8	8	8	8
Test 1.1.10 - RPC	9	8	8	9	9
Test 1.1.11 - SSH	1	1	1	1	1
Test 1.1.12 - Telnet	1	1	1	1	1
Test 1.1.13 - Database	1	1	1	1	1
Test 1.1.14 - Mail	1	1	1	1	1
Test 1.1.15 - Voice	1	0	0	1	1
Total	**110**	**105**	**105**	**107**	**107**
		95%	**95%**	**97%[2]**	**97%[2]**

Test 1.2 – Resistance to False Positives	Default	Custom
Test 1.2.1 - Suspicious FTP traffic	PASS	PASS
Test 1.2.2 - HTTP “exploit” using incorrect method	PASS	PASS
Test 1.2.3 - Retrieval of Web page containing “suspicious” URLs	PASS	PASS
Test 1.2.4 - Simple SMTP QUIT command	PASS	PASS
Test 1.2.5 - Normal NetBIOS copy of “suspicious” files	FAIL	PASS
Test 1.2.6 - Normal NetBIOS traffic	FAIL	PASS
Test 1.2.7 - POP3 e-mail containing “suspicious” URLs	PASS	PASS
Test 1.2.8 - POP3 e-mail with “suspicious” DLL attachment	PASS	PASS
Test 1.2.9 - POP3 e-mail with “suspicious” Web page attachment	PASS	PASS
Test 1.2.10 - SMTP e-mail transfer containing “suspicious” URLs	PASS	PASS
Test 1.2.11 - SMTP e-mail transfer with “suspicious” DLL attachment	PASS	PASS
Test 1.2.12 - SMTP e-mail transfer with “suspicious” Web page attachment	PASS	PASS
Test 1.2.13 - SNMP V3 packet with invalid parameter	PASS	PASS
Test 1.2.14 - Fake DNS /bin/sh buffer overflow	FAIL	PASS
Test 1.2.15 - Inter-firewall communication traffic	PASS	PASS
Test 1.2.16 - Fake SQL Slammer traffic	PASS	PASS
Test 1.2.17 - File copy of GIF file (contains bytes which look like NOP sled)	PASS	PASS
Total Passed	**14 / 17**	**17 / 17**

Section 2 - IPS Evasion

Test 2.1 – Evasion Baselines	Detected?	Blocked?
Test 2.1.1 - NSS Back Orifice ping	YES	YES
Test 2.1.2 - Back Orifice connection	YES	YES
Test 2.1.3 - FTP CWD root	YES	YES
Test 2.1.4 - ISAPI printer overflow	YES	YES
Test 2.1.5 - Showmount export lists	YES	YES
Test 2.1.6 - Test CGI probe (/cgi-bin/test-cgi)	YES	YES
Test 2.1.7 - PHF remote command execution	YES	YES
Total	**7 / 7**	**7 / 7**

Test 2.2 – Packet Fragmentation/Stream Segmentation	Detected?	Decoded?	Blocked?
Test 2.2.1 - IP fragmentation - ordered 8 byte fragments	YES	YES	YES
Test 2.2.2 - IP fragmentation - ordered 24 byte fragments	YES	YES	YES
Test 2.2.3 - IP fragmentation - out of order 8 byte fragments	YES	YES	YES
Test 2.2.4 - IP fragmentation - ordered 8 byte fragments, duplicate last packet	YES	YES	YES
Test 2.2.5 - IP fragmentation - out of order 8 byte fragments, duplicate last packet	YES	YES	YES
Test 2.2.6 - IP fragmentation - ordered 8 byte fragments, reorder fragments in reverse	YES	YES	YES
Test 2.2.7 - IP fragmentation - ordered 16 byte fragments, fragment overlap (favour new)	YES	YES	YES
Test 2.2.8 - IP fragmentation - ordered 16 byte fragments, fragment overlap (favour old)	YES	YES	YES
Test 2.2.9 - TCP segmentation - ordered 1 byte segments, interleaved duplicate segments with invalid TCP checksums	YES	YES	YES
Test 2.2.10 - TCP segmentation - ordered 1 byte segments, interleaved duplicate segments with null TCP control flags	YES	NO	YES
Test 2.2.11 - TCP segmentation - ordered 1 byte segments, interleaved duplicate segments with requests to resync sequence nos. mid-stream	YES	YES	YES
Test 2.2.12 - TCP segmentation - ordered 1 byte segments, duplicate last packet	YES	YES	YES
Test 2.2.13 - TCP segmentation - ordered 2 byte segments, segment overlap (favour new)	YES	YES	YES
Test 2.2.14 - TCP segmentation - ordered 1 byte segments, interleaved duplicate segments with out-of-window sequence numbers	YES	YES	YES
Test 2.2.15 - TCP segmentation - out of order 1 byte segments	YES	YES	YES
Test 2.2.16 - TCP segmentation - out of order 1 byte segments, interleaved duplicate segments with faked retransmits	YES	YES	YES
Test 2.2.17 - TCP segmentation - ordered 1 byte segments, segment overlap (favour new)	YES	YES	YES
Test 2.2.18 - TCP segmentation - out of order 1 byte segments, PAWS elimination (interleaved dup segments with older TCP timestamp options)	YES	YES	YES
Test 2.2.19 - IP fragmentation - out of order 8 byte fragments, interleaved duplicate packets scheduled for later delivery	YES	YES	YES
Test 2.2.20 - TCP segmentation - ordered 16 byte segments, segment overlap (favour new (Unix))	YES	YES	YES
Total	**20 / 20**	**19 / 20**	**20 / 20**

Test 2.3 – URL Obfuscation	Detected?	Decoded?	Blocked?
Test 2.3.1 - URL encoding	YES	YES	YES
Test 2.3.2 - /./ directory insertion	YES	YES	YES
Test 2.3.3 - Premature URL ending	YES	YES	YES
Test 2.3.4 - Long URL	YES	YES	YES
Test 2.3.5 - Fake parameter	YES	YES	YES
Test 2.3.6 - TAB separation	YES	YES	YES
Test 2.3.7 - Case sensitivity	YES	YES	YES
Test 2.3.8 - Windows \ delimiter	YES	YES	YES
Test 2.3.9 - Session splicing	YES	YES	YES
Total	**9 / 9**	**9 / 9**	**9 / 9**

Test 2.4 – Miscellaneous Obfuscation Techniques	Detected?	Decoded?	Blocked?
Test 2.4.1 - Altering default ports	NO	NO	NO
Test 2.4.2 - Inserting spaces in FTP command lines	NO	NO	NO
Test 2.4.3 - Inserting non-text Telnet opcodes in FTP data stream	NO	NO	NO
Test 2.4.4 - Polymorphic mutation (ADMmutate)	YES	YES	YES
Test 2.4.5 - Altering protocol and RPC PROC numbers	YES	YES	YES
Test 2.4.6 - RPC record fragging (MS-RPC and Sun)	YES	YES	YES
Test 2.4.7 - HTTP exploits to port <> 80	YES	YES	YES
Total	**4 / 7**	**4 / 7**	**4 / 7**

Section 3 - Stateful Operation

Test 3.1 – Stateless Attack Replay	Alert?	Blocked?	Pass/Fail
Test 3.1.1 - Stateless Web exploits	NO	NO	PASS
Test 3.1.2 - Stateless FTP exploits	NO	NO	PASS

Test 3.2 – Simultaneous Open Connections (default settings)							
Number of open connections	10,000	25,000	50,000	100,000	250,000	500,000	1,000,000
Test 3.2.1 - Attack Detection	PASS	PASS	PASS	PASS	FAIL	FAIL	FAIL
Test 3.2.2 - Attack Blocking	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Test 3.2.3 - State Preservation	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Test 3.2.4 - Legitimate traffic blocking	PASS	PASS	PASS	PASS	FAIL	FAIL	FAIL

Test 3.3 – Simultaneous Open Connections (after tuning)							
Number of open connections	10,000	25,000	50,000	100,000	250,000	500,000	1,000,000
Test 3.3.1 - Attack Detection	PASS	PASS	PASS	PASS	PASS	PASS[3]	FAIL
Test 3.3.2 - Attack Blocking	PASS	PASS	PASS	PASS	PASS	PASS[3]	PASS
Test 3.3.3 - State Preservation	PASS	PASS	PASS	PASS	PASS	PASS[3]	PASS
Test 3.3.4 - Legitimate traffic blocking	PASS	PASS	PASS	PASS	PASS	PASS[3]	FAIL

Section 4 - Detection/Blocking Performance Under Load

Test 4.1 – UDP traffic to random valid ports		125Mbps	250Mbps	375Mbps	500Mbps	Max
Test 4.1.1 - 256 byte packet test - max 226,500pps	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	
Test 4.1.2 - 550 byte packet test - max 110,000pps	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	
Test 4.1.3 - 1514 byte packet test - max 61,000pps	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	

Test 4.2 – HTTP "maximum stress" traffic with no transaction delays		125Mbps	250Mbps	375Mbps	500Mbps	Max
Test 4.2.1 - Max 1250 connections per second - ave packet size 1000 bytes - max 60,000 packets per second	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	
Test 4.2.2 - Max 2500 connections per second - ave packet size 540 bytes - max 112,500 packets per second	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	
Test 4.2.3 - Max 5000 connections per second - ave packet size 440 bytes - max 137,500 packets per second	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	
Test 4.2.4 - Max 10000 connections per second - ave packet size 360 bytes - max 160,000 packets per second	Detected	100%	100%	100%	N/A[4]	400Mbps
	Blocked	100%	100%	100%	N/A[4]	

Test 4.3 – HTTP "maximum stress" traffic with transaction delays		125Mbps	250Mbps	375Mbps	500Mbps	Max
Test 4.3.1 - Max 2500 connections per second - ave packet size 540 bytes - max 112,500 packets per second - 10 sec delay - max 25,000 open connections	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	
Test 4.3.2 - Max 5000 connections per second - ave packet size 440 bytes - max 137,500 packets per second - 10 sec delay - max 50,000 open connections	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	

Test 4.4 – Protocol mix		125Mbps	250Mbps	375Mbps	500Mbps	Max
Test 4.4.1 - 72% HTTP (540 byte packets) + 20% FTP + 6% UDP (256 byte packets). Max 2000 connections per second - ave packet size 540 bytes - max 107,500 packets per second - max 375 open connections	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	

Test 4.5 – Real World traffic		125Mbps	250Mbps	375Mbps	500Mbps	Max
Test 4.5.1 - Pure HTTP (simulated browsing session on NSS Web site). Max 2350 connections per second - 10 new users per second - ave packet size 560 bytes - max 105,000 packets per second	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	
Test 4.5.2 - Protocol mix - 72% HTTP (simulated browsing sessions as 2.5.1) + 20% FTP + 6% UDP (256 byte packets). Max 1850 connections per second - ave packet size 560 bytes - max 102,500 packets per second - max 750 open connections	Detected	100%	100%	100%	100%	500Mbps
	Blocked	100%	100%	100%	100%	

Section 5 - Latency & User Response Times

Test 5.1 – Latency	Packet Size	125Mbps	250Mbps	375Mbps	500Mbps
Test 5.1.1 Average latency (µs) with no background traffic	256	86.34	88.00	89.19	91.13
	550	111.15	112.04	112.48	114.78
	1000	138.26	140.16	142.68	145.21
Test 5.1.2 Average latency (µs) with background traffic (250Mbps HTTP traffic, max 1250 connections per second - ave packet size 540 bytes - max 56,250 packets per second)	256	133.03			
	550	149.58			
	1000	175.27			
Test 5.1.3 Average latency (µs) when under attack (50Mbps SYN flood (74,000cps))	256	N/A[5]			
	550	N/A[5]			
	1000	N/A[5]			

Test 5.2 – User Response Times	Attempted Trans	Failed Trans	Min Page Response	Max Page Response	Ave Page Response
Test 5.2.1 - Web page response (ms) with no background traffic (250Mbps HTTP traffic, max 1250 connections per sec - ave packet size 540 bytes - max 56,250 packets per sec)	781149	0	201	215	203
Test 5.2.2 - Web page response (ms) when under attack (250Mbps HTTP traffic, max 1250 connections per sec - ave packet size 540 bytes - max 56,250 packets per sec PLUS 50Mbps SYN flood (74,000cps))	N/A[5]	N/A[5]	N/A[5]	N/A[5]	N/A[5]

Section 6 - Stability & Reliability

Test ID	Result
Test 6.1.1 - Blocking Under Extended Attack	100%
Test 6.1.2 - Passing legitimate traffic under extended attack	100%
Test 6.1.3 - ISIC/ESIC/TCPSIC/UDPSIC/ICMPSIC	PASS

Section 7 - Management Interface

Test ID	Result
Test 7.1.1 - Open Ports	PASS
Test 7.1.2 - ISIC/ESIC/TCPSIC/UDPSIC/ICMPSIC	PASS
Test 7.1.3 - ISIC attacks detected against management interface?	NO

Notes:

1. Missing test cases were detected successfully when HTTP server-to-client signatures were enabled. HTTP server-to-client signatures were disabled for performance reasons
2. 100 per cent recognition rate is possible when HTTP server-to-client signatures are enabled
3. Maximum open TCP connections supported officially is 220,000. 500,000 was achieved by reducing memory available for non-TCP “flows”
4. This test could not be completed - maximum connections per second is approx 8,000 (400Mbps)
5. This test could not be completed - SYN Protector was not capable of handling 50Mbps of SYN Flood traffic used in our test - device was overwhelmed and all traffic blocked. Recommend IDP 600F is deployed behind an attack mitigation device

Section 1: Detection Engine

We installed one IDP 600F sensor with the latest signature set. The Security Policy was created from scratch, switching on all Attack Objects including *Critical*, *High, Medium* and *Low* (omitting only *Informational* signatures, which are more suited to auditing purposes), as well as turning on *Backdoor Protection* and the *SYN Protector*. We also enabled fragmented packet detection and flow error detection in order to identify certain DOS and ICMP attacks.

Finally, we disabled all HTTP server-to-client signatures for performance reasons (this is achieved via a single sensor setting, rather than having to modify the Security Policy - a useful feature). Note that this would be a valid configuration for a purely internal deployment where HTTP server-to-client signatures are largely irrelevant in most situations, and where performance is paramount.

There is a severe performance impact when enabling HTTP server-to-client signatures, reducing throughput of the device to 200-250Mbps and making it more suitable for a perimeter deployment (where these signatures would be essential).

Signature recognition (with blocking disabled) was excellent out of the box at 95 per cent. This figure was increased to 97 per cent following a signature pack update which Juniper provided within 48 hours. Blocking performance was identical throughout the tests. Note that with the HTTP server-to-client signatures enabled, the attack recognition rose to a perfect 100 per cent (though with a corresponding decrease in maximum performance as a result).

All of our “false negative” (modified exploit) cases were detected correctly out of the box, demonstrating that the Juniper signatures are generally designed to detect the underlying vulnerability rather than a specific exploit.

With some of our more complex test cases we noted an occasional tendency for the IDP 600F to raise several alerts for a single exploit. This is generally as a result of exploits which trigger both protocol errors and pattern matches within the same attack. Whilst we could not fault the accuracy of those alerts, it would be nice to see them correlated or aggregated somehow in the GUI, allowing the administrator to focus on a single “incident” and drill down to see the multiple alerts beneath when required.

A major concern in deploying an IPS is the blocking of legitimate traffic, and the IDP 600F did block three of our false positive test cases out of the box. This was rectified following the signature update, after which we noted no further false positives throughout the rest of the testing.

The IDP 600F comes with a number of suggested Protection Policy templates to aid the administrator when setting up initially - these include suggested policies for file servers, Web servers, DNS servers, and so on, along with a typical “default” policy.

These are based on a number of carefully constructed Dynamic Groups which will allow Juniper to modify these policies with each signature update, adding new signatures automatically, or moving signatures between *alert* and *block* modes as confidence in a new signature increases.

Section 2: IPS Evasion

Resistance to known evasion techniques was excellent, with the IDP 600F achieving a clean sweep across the board in all our main evasion tests. *Fragroute, Whisker, ADMmutate* and even *RPC record fragging* all failed to trick the IDP 600F into ignoring valid attacks.

Not only were the fragmented and obfuscated attacks blocked successfully, but all but one of them - a single *fragroute* evasion - were decoded accurately as well when not blocking. Only a couple of miscellaneous FTP evasion techniques were effective in evading the device - although we do not consider this to be too serious, it is a situation which we would prefer to see remedied as soon as possible, and Juniper is working on a solution at the time of writing.

Section 3: Stateful Operation

Out of the box, the device maintained state on up to 100,000 open connections, successfully detecting our half-open exploit as it was completed. It also continued to detect and block new exploits as we maintained 100,000 open connections, and no legitimate traffic was blocked during this stage of these tests.

With a simple change to the flow-related parameters in the GUI, it was possible to maintain state across 500,000 open connections (though Juniper actually recommends 220,000 as the maximum on this device).

Default operation of the device is to refuse new connections when the state tables are full or resources are low, meaning that legitimate traffic is blocked and new attacks are not alerted. However, state is maintained completely on existing flows and attack traffic should never be allowed through the device.

Our extensive testing initially uncovered a problem when maximum connections was configured at 500,000 and then that number was exceeded, completely exhausting available memory. In this corner-case situation (bear in mind the supported maximum is 220,000 open connections), a default setting in the sensor caused traffic to be passed uninspected as memory was exhausted. This setting was changed during testing to ensure all traffic was blocked in this situation, and the change should be made permanent with the next product update.

Stateless “exploits” are not alerted upon (this is correct behaviour in order to be resistant to *Stick* and *Snot* tools), and mid-flows are blocked by default. It is not possible to configure the device to allow mid-flows.

Section 4: Detection/Blocking Performance Under Load

Note that the Juniper Networks IDP 600F was tested as a 500Mbps IPS device.

Performance at almost all levels of our load tests was excellent, with 100 per cent of all attacks being detected and blocked under most load conditions.

The one exception was Test 4.2.4 where we determined that there was a limit of approximately 8,000 HTTP connections per second (equating to approximately 400Mbps) meaning the test could not be completed. Beyond this limit, legitimate connections began to fail, but all attack traffic was still blocked successfully.

However, we would happily confirm Juniper's 500Mbps rating for this device under normal network conditions, and would actually consider it to be conservative on a typical network.

Section 5: Latency & User Response Times

Basic latency figures were very good for a device of this type at all traffic loads and packet sizes, ranging from 86µs with 125Mbps of 256 byte packets, to 145µs with 500Mbps of 1000 byte packets.

Behaviour through all of the tests with no background traffic was very predictable, with relatively small increases in latency as traffic levels increased from 125Mbps to 500Mbps across each packet size.

Placing the device under a half load of 250Mbps of HTTP traffic we noted small increases in latency across the board, rising from 86µs to 133µs with 250 byte packets, from 111µs to 149µs with 550 byte packets, and from 138µs to 175µs with 1000 byte packets.

These results are excellent for a 500Mbps device, and given that HTTP response times were also excellent, we believe that the IDP 600F could be deployed anywhere on a 500Mbps network, either internally or at the perimeter.

Whether or not the SYN Protector was enabled, the IDP 600F was unable to handle the 50Mbps of SYN flood traffic we launched during our DOS/DDOS tests. Once we had exceeded the 8000 connections per second (i.e. 8000 SYNs per second) maximum we discovered in the HTTP stress tests, the device became unresponsive, effectively blocking all traffic, both malicious and legitimate.

More effective SYN protection is planned for a future release, but for now we recommend the IDP 600F is deployed behind an attack mitigation device or firewall with full DOS/DDOS mitigation capabilities.

Section 6: Stability & Reliability

The IDP 600F performed consistently and completely reliably throughout our tests. Under eight hours of extended attack (comprising millions of exploits mixed with genuine traffic) it continued to block 100 per cent of attack traffic, whilst passing 100 per cent of legitimate traffic.

Exposing the sensor interface to ISIC-generated traffic had no adverse effect, and the device continued to detect and block all other exploits (as well as raising ISIC-related alerts) throughout and following the ISIC attack.

Section 7: Management Interface

Open ports on the management interface are restricted to TCP/22 (SSH) and UDP/7201-2 (management communications), but none of these are visible to port scanners.

The extended ISIC attack against the management interface had no effect on the appliance and its ability to detect and block attacks, and communication between sensor and management server was barely affected. No alerts were raised during the attack.

The sensor continued to work perfectly throughout and following the ISIC attack, and there were no residual stability problems.