Intoto IntruPro V3.0

Technical Evaluation

An NSS Group Report

First published June 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*.

INTOTO INTRUPRO V3.0

Executive Summary

Intoto delivers a software-based IPS product, IntruPro, licensed directly to OEMs. This allows them quickly to integrate IPS technology in their own hardware platform in order to build UTM devices, security appliances or security switches. The architecture is agnostic to hardware platforms, operating systems and hardware accelerators. Intoto submitted a reference hardware platform for the testing.

Due to the fact that Intoto provides the IntruPro software platform to be integrated with the OEM's hardware, OS and any other existing software components, it is very important for the OEM to tune and optimise their product appropriately. The efficacy of the product therefore depends to a certain extent on the skill of the integrator, and it should be understood that *NSS Approval* for IntruPro as a software product does not automatically confer *NSS Approval* on those devices using it. Potential customers should therefore ensure that any vendor offering an appliance based on the IntruPro product has obtained specific *NSS Approval* for that device before purchasing.

The product is available in both layer 3 (routed) and layer 2 (transparent bridge) versions, making it suitable for both perimeter and core network deployments. On the reference hardware provided for this test (dual Xeon processor, 2GB RAM, and dual copper Gigabit ports) IntruPro provided itself capable of handling 100Mbps of traffic under normal network conditions.

Blocking rates were good out of the box, improving to 95 per cent following an update, and resistance to common IDS/IPS evasion techniques was excellent. We did, however, notice a tendency for the device to raise multiple alerts for a single exploit, and false positives could be a concern on some networks. Throughput and latency were good under all network loads and across all packet sizes up to the rated 100Mbps. We also found IntruPro to be very stable and reliable under extended attack, with effective SYN flood mitigation capabilities.

Management and configuration of single IntruPro devices is straightforward via the three-tier management architecture provided. A lack of any centralised policy management, however, means that the current solution will not scale well, forcing the administrator to attach to each Sensor individually in order to configure it.

Alert handling and reporting are adequate, offering a range of filtering options on the log entries, and the ability to create custom alerts. Unfortunately, having created customised report filters and layouts, it is not possible to save these for re-use.

In general, we would conclude that configuration and alert handling are acceptable for a single device or low number of devices, but not currently suitable for larger-scale deployments. Intoto also allows OEM vendors to integrate with their existing management software or a third party CMS tool.

Architecture

Intoto offers a three-tier architecture for sensor management - there is no direct Sensor management capability in the current version.

Although this clearly provides reliability and scalability, it does mean that a separate, dedicated host is always required for the management server component, possibly making it a less attractive solution for only one or two sensors.

Intoto also allows the OEM vendors to integrate their existing management software or web based interface using APIs provided by IntruPro sensor. The main components of the IntruPro system are:

- ***Database Store -*** *Stores all data, logs, configuration settings, etc. MySQL and PostGreSQL are supported.*
- ***Log Listener -*** *Daemon which polls all Sensors and collects log information, storing it in the Database Store. Depending on performance/scalability requirements, this can run on a dedicated machine, on the same machine as the Database Store, or even on the Sensor.*
- ***Console -*** *This is the Java-based management GUI which allows the administrator to analyse, view and modify the data present in the Database Store. Since the Console and Log Listener daemon are de-coupled, it is possible to run multiple Consoles looking into the same Database Store.*
- ***Sensor -*** *The Sensor software can run on a range of operating system and hardware platforms to provide in-line IPS protection. A maximum of 16 Sensors can be managed from a single management system. Communications between the Sensor and Log Listener can be encrypted if required (a pre-shared secret is provided by the administrator during initial configuration).*

The first three components are collectively referred to as the *IntruPro Manager*.

IntruPro internally uses some components of its layer 3 firewall for layer 3 and 4 attack detection and prevention. The product submitted for testing was tested for routed gateway mode, making it most suitable for perimeter deployments.

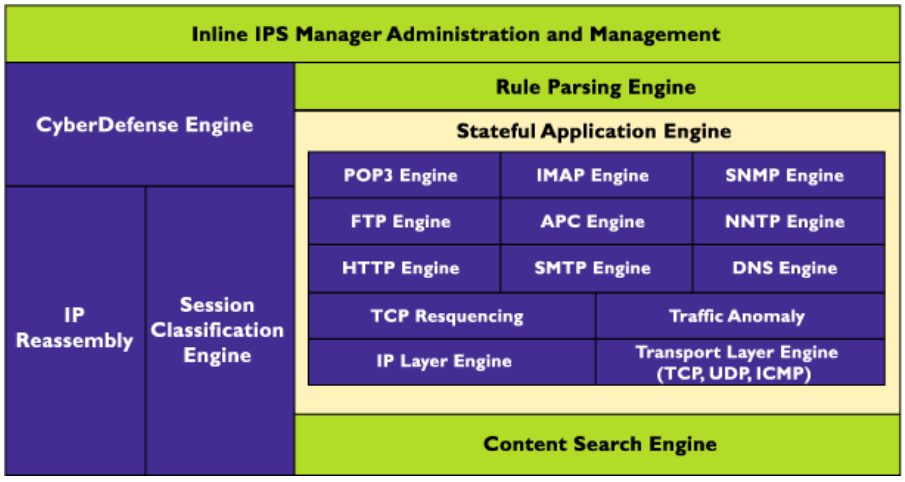

Figure 1 - IntruPro: Sensor Architecture

A layer 2 transparent bridge version is also available along with the layer 2 Firewall offered by Intoto. Intoto has a history of application-layer firewall development, and therefore it is not surprising to see a wide array of protocol decoders (called Application Engines) in the IPS product

This allows the IntruPro Sensor to make use of a range of detection techniques including:

- *Signature based - using both fixed patterns and regular expressions*
- *Protocol anomaly*
- *Traffic Anomaly*

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.

Note that Intoto IntruPro is a software product and was tested on reference hardware (dual Xeon processor, 2GB RAM, copper Gigabit ports) as a 100Mbps IPS device.

Performance at almost all levels of our load tests was impeccable, with 100 per cent of all attacks being detected and blocked under all load conditions except at the 2000 connections per second level (maximum connection rate was around 1400cps - see *Test Results* section for full details)

We would happily confirm Intoto's 100Mbps rating for this device under normal network conditions, however.

The basic latency figures of IntruPro on the reference hardware were good for a 100Mbps device across the board under all traffic loads. They ranged from 157µs with 25Mbps of 256 byte packets, to 222µs with 100Mbps of 1000 byte packets.

Behaviour throughout the tests with no background traffic was reasonably consistent and predictable, with small decreases as additional network load was applied from 25Mbps to 100Mbps. Placing the device under a half load of 50Mbps of HTTP traffic increased latency somewhat, rising from 157µs to 220µs with 256 byte packets, from 214µs to 243µs with 550 byte packets, and from 254µs to 289µs with 1000 byte packets.

HTTP response times were also very good, demonstrating an average of 203ms with 50Mbps of HTTP traffic.

Latency when under 10Mbps (14800 packets per second) of SYN flood traffic was excessive due to packet loss. Despite this, the SYN flood mitigation technique (SYN proxy) used by Intoto was very effective, with very little of the DOS traffic making its way through to the target server. Although average HTTP response time for legitimate traffic increased to 623ms during the attack, there were very few failed transactions,.

IntruPro 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 on the device at any time, and relevant protocol anomaly alerts were raised. All other malicious traffic continued to be blocked successfully during the attack, and there were no residual stability problems once the ISIC attack had been terminated.

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

Security Effectiveness

We installed one sensor with the latest signature library, and enabled **all** signatures, with traffic monitoring in both directions. Out of the box, blocking performance was good at 80 per cent, and was improved to 95 per cent following the application of a signature update after 48 hours.

Detection/recognition rate was slightly lower (89 per cent following update) due to the fact that many DOS and ICMP attacks are blocked without alerting by the firewall module. We also noted that many of our test cases raised several alerts for a single exploit. This can be annoying, and we would prefer that the product performed simple correlation to reduce the number of superfluous alerts raised. It can, however, be mitigated by the fact that the source aggregation feature of the management interface results in a single entry on the GUI which can then be expanded to show the individual alerts if required.

Performance in our “false negative” tests was good out of the box, and improved following the signature update, leaving just a single miss out of 14.

A major concern in deploying an IPS is the blocking of legitimate traffic. The device initially failed five test cases in our false positive test suite, and although all of these were eliminated following the signature update, we also noted several false positives during our FTP testing, the result of some sloppy port-based backdoor signatures. Overall, false positives would be a concern for us with this device at present, and we would encourage potential purchasers to test in their own network before buying.

Resistance to known evasion techniques was excellent, with IntruPro achieving a clean sweep across the board in almost all of our evasion tests. *Fragroute* and *Whisker* both failed to deceive the device into ignoring valid attacks, and all but one of the attempts were decoded accurately. Some minor configuration was required to enforce blocking for two of the test cases, but this should become the default configuration in future releases.

Of the miscellaneous evasion techniques, only changing ports on backdoors caused problems (a not untypical situation, since it is expensive on resources to monitor for all backdoors on all ports), but the rest (including extensive RPC record fragging) were handled well.

Out of the box, Intoto claims that the IntruPro on 100Mbps reference hardware can handle up to 50,000 open connections. This was not quite confirmed in testing, the device reaching its limits at around 45,000 open connections. We consider this to be just adequate for a 100Mbps device.

During testing, we verified IntruPro’s ability to handle up to 45,000 simultaneously open connections whilst successfully maintaining state on our half-open exploits.

The default operation of the device is to reject new connections when the state tables are full or resources are low, and this means that once the connection limit is exceeded, the device continues to maintain state on existing connections (thus detecting our half-open exploits), but inevitably, as a consequence, begins to block legitimate traffic. This behaviour is not configurable at run time.

Stateless “exploits” are blocked by default, with no alerts. This behaviour is not configurable at run time, and there is no grace period following device start-up during which existing connections (which will appear as mid-flows initially) will be passed successfully.

Please refer to the *Testing Methodology* section for full details of the methodology used and complete 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

The IntruPro Sensor would normally be supplied as a turnkey appliance with a hardened and tuned operating system running the Sensor software. It is important that the vendor performs this task carefully, since it can affect both security and performance of the finished appliance.

During testing, we noted several OS-specific and firewall-specific tuning parameters required amending in order to complete several of our tests. Intoto is now aware of these and will hopefully provide guidance for vendors offering their software in appliance form.

All that is generally required when installing the Sensor is to configure the management interface and IP address, and specify the address of the management server. All of this is performed via the extensive text-based *Command Line Interface* (CLI) directly on the Sensor console. This is also the place to perform any additional performance tuning which may be required.

Installation of the management server software is on a dedicated Windows (NT, 2000, XP, 2003) box, and requires that both the latest Java Run-time Environment (JRE) and database (MySQL or PostGreSQL) software is installed first.

Once installed, a simple set-up Wizard guides the administrator through the initial configuration tasks, such as specifying database parameters, configuring the Log Listener daemon, creating Sensors, creating administrator accounts, and performing the first signature update.

Documentation is good, with a *Log Listener Administration Guide* and *IntruPro Manager Administration Guide* provided in both electronic (PDF) and hard copy versions.

Configuration

All day-to-day configuration is performed via the IntruPro Console.

The main screen is divided into four parts: the *Main Toolbar, Top Toolbar, Side Toolbar,* and *Central Area* (which acts as the main viewing and data-entry area).

The operation of the Console follows a Web browser paradigm where there is a *Back* button which can be used to step backwards through screens previously accessed - this can make it quick to navigate when drilling down several levels.

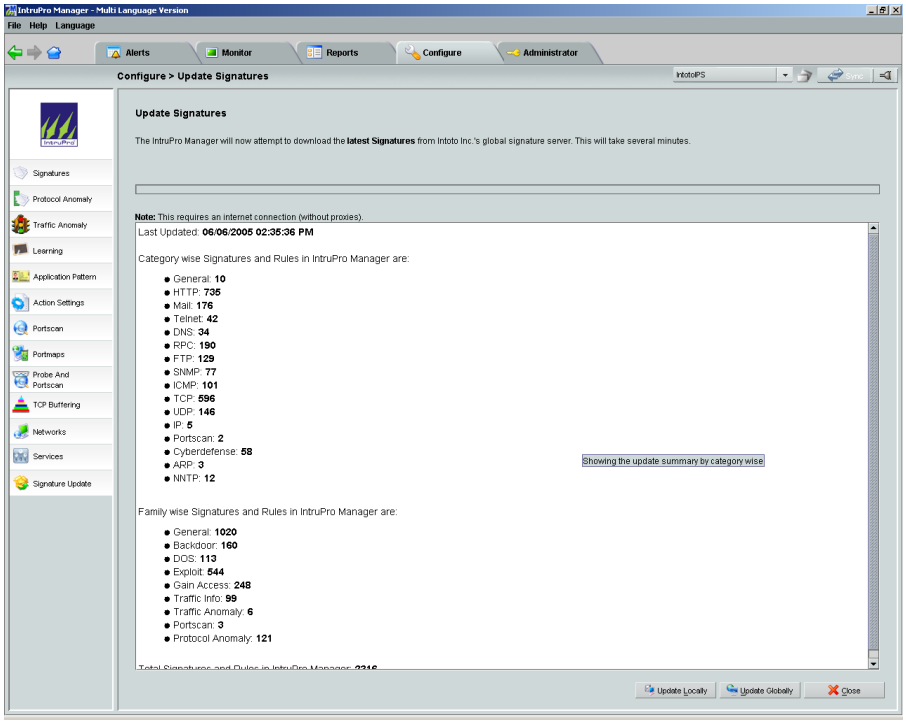

Figure 2 - IntruPro: The IntruPro Console

The application options are grouped into tabbed entries to the main toolbar, and as each tab is selected, sub-options are displayed in the side toolbar.

The main options available are:

- ***Alerts -*** *Create and view custom alerts*
- ***Monitor -*** *Real-time graphical monitoring of malicious activity and per-protocol statistics*
- ***Report -*** *Access alert logs*
- ***Configure -*** *Sensor configuration*
- ***Administrator -*** *Maintenance tasks*

All configuration tasks apply to the currently selected Sensor only, which is chosen from a drop-down menu of available Sensors at the top of the screen. In the current release, it is not possible to manage more than one Sensor at a time.

Multiple Sensors can, however, be managed individually from a single Console, and new Sensors can be added via the *Administrator* tab simply by specifying the IP address of the management interface and a pre-shared secret if encryption is used.

The Administrator tab also provides the means to configure database access, SMTP server parameters for e-mail alerts, Log Listener settings, and administrator user name and password. Note that there is no role-based access and only a single administrator account in the IntruPro Manager, and that user has access to all features of the GUI once logged in. Whilst acceptable for single-device management, this is not a system which is suitable for large-scale enterprise deployments without additional CMS integration.

Once a configuration is completed, it can be saved to the database, and synchronised to the Sensor using the *Sync* button at the top of the screen. During the synchronisation process, the Sensor is unavailable for a short period of time (packets will be passed without inspection).

Policy Management

Within IntruPro Manager there is no concept of policies which can be defined, saved and then applied to multiple Sensors. Instead, each Sensor is configure directly and independently - it is not even possible to upload the configuration from one Sensor and apply it to another, which could make large scale deployments cumbersome and error-prone. This is not currently a scalable management solution.

For single-device management, however, it is fine, and configuring a Sensor is very straightforward via the *Configuration* tab. Selecting this tab (after selecting the Sensor to be configured from the drop-down menu) brings up a list of configuration sub-options in the side toolbar, the most important being the *Signatures* option.

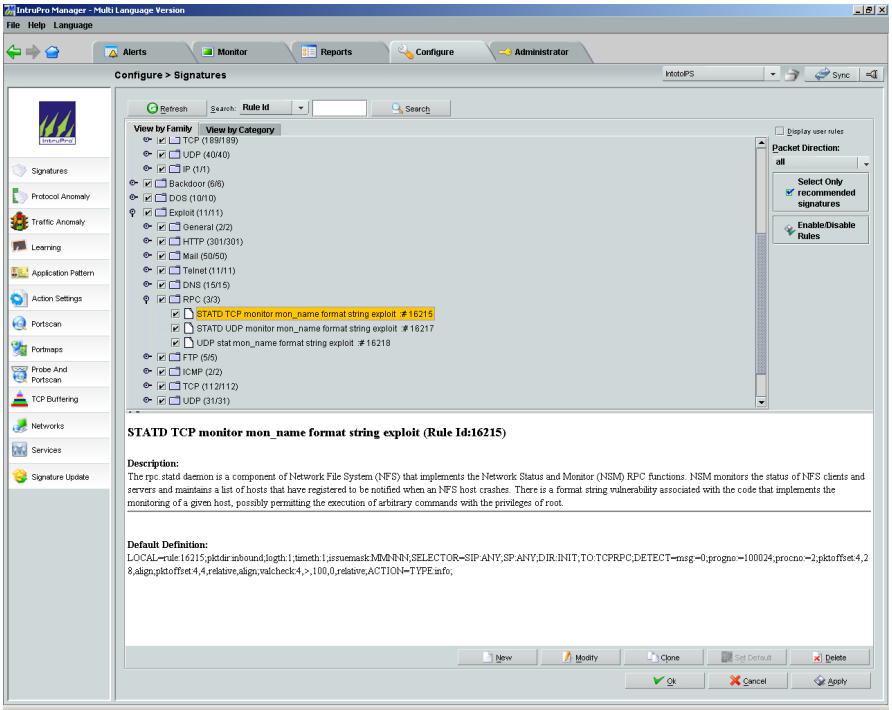

Figure 3 - IntruPro: Configuring Signatures

The signature database is divided into both *Families* (*General, Backdoor, DOS, Gain Access, Exploit, Traffic Info*) and *Categories* (*General, HTTP, Mail, RPC, FTP, Telnet, DNS, SNMP, ICMP, TCP, UDP, NNTP, IP*) and entire Families or Categories can be enabled or disabled by selecting the check box against each in the hierarchical menu tree.

Naturally, on selecting a specific Category/Family, the individual signatures within can also be enabled/disabled.

Selecting an individual signature will display the description in the lower pane, and it is possible to *enable/disable, clone* or *delete* the signature using buttons along the bottom of the screen. All of the signature detail is visible when editing or cloning an existing signature, making it easy to tune any of the built-in signatures as required.

It is also possible to create signatures from scratch, and all custom signatures (both those created from scratch and any built-in signatures which have been modified) can be examined via a separate *User Rules* menu tree (which will remain untouched during future signature update operations).

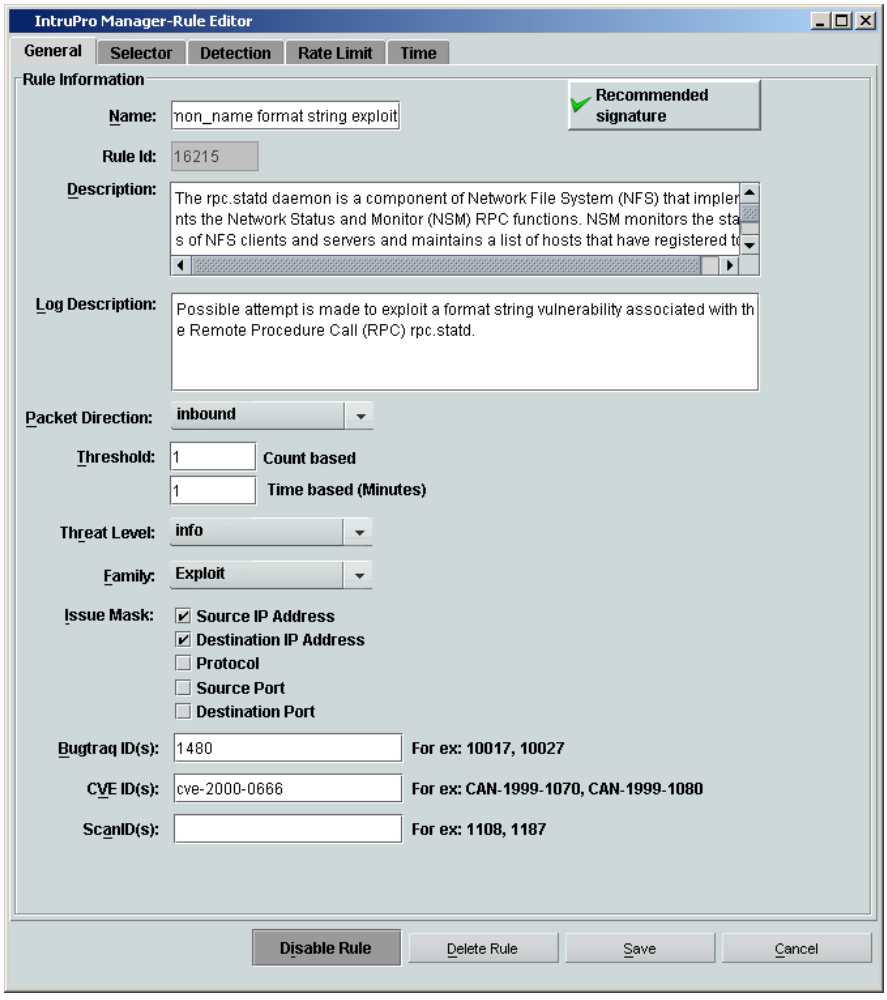

Figure 4 - IntruPro: View Signature details

Full access is provided to all of the fields in all of the protocol decoders (as well as simple pattern matching fields) which can make signature creation somewhat complex for those new to the task, though obviously very powerful for those with more experience.

Obviously the ability to clone existing signatures and then modify them should help tremendously in the creation of new ones, and the accompanying documentation is also very good.

A very useful search facility allows the administrator to enter simple search criteria (for example, the text "*IIS*" or "*Apache*") and have all the signatures matching that criteria extracted and displayed in a separate window. One or more of those signatures can then be selected via the check box next to each one, and opened for editing via a single click.

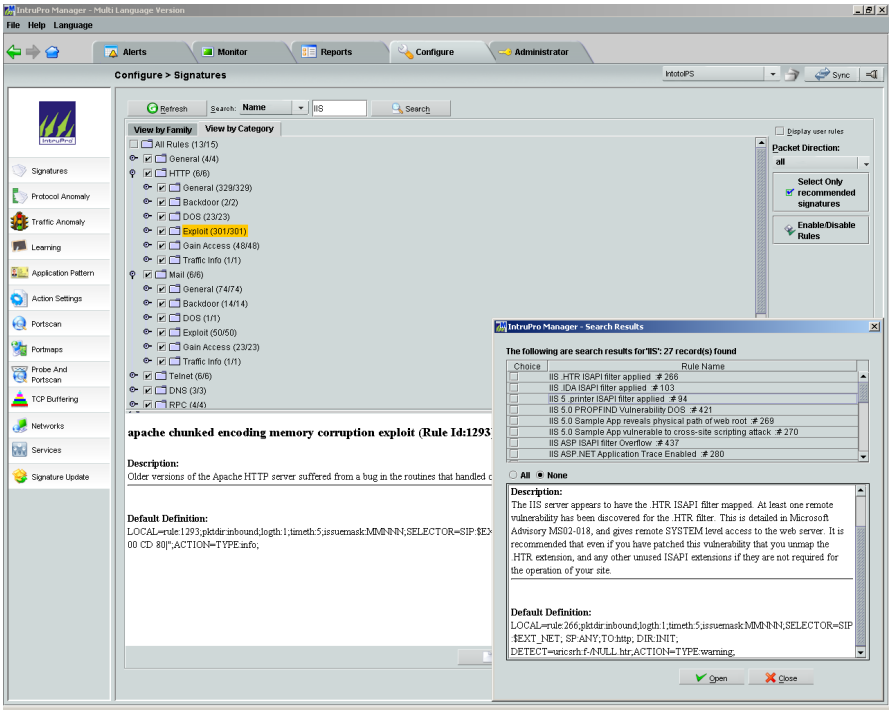

Figure 5 - IntruPro: Signature search results

There are three more very important buttons on the *Signature* screen:

- ***Packet Direction -*** *allows the administrator to select whether to monitor inbound traffic only, outbound traffic only, or both directions. We had selected "Both" for testing, since this provides the most complete security coverage, but monitoring only inbound traffic would still provide excellent coverage whilst probably allowing the device to monitor slightly more than the 100Mbps of traffic tested*
- ***Enable/Disable Signatures -*** *brings up a single list of all signatures along with their current enabled/disabled status. Each signature has a check box alongside indicating "enabled" and allowing the administrator to quickly enable or disable individual signatures directly*
- ***Select Only Recommended Signatures -*** *the most useful button for new deployments. Every Intoto signature has a "Recommended" flag which indicates the confidence the signature writers have in terms of accuracy, resistance to false positives, and so on. Selecting this option enables only those Recommended signatures, and provides a good initial deployment option*

Actions taken when signatures are matched with malicious traffic are determined by the *Action Settings* tab:

- ***Disconnect Session -*** *Drop malicious packet, mark the rest of the session "bad", and send TCP Resets to source and destination IPs*
- ***Drop Packet -*** *Drop malicious packet only*
- ***Report Only -*** *Write log entry, but take no blocking action*
- ***Ignore -*** *Pass traffic with no alert and no blocking action*

We would prefer to see slightly more granularity here - for example, the ability to drop a session (drop packet and mark the rest of the session bad) but **not** send TCP Resets.

Action Settings are applied to each of the *Threat Levels* (*Critical, Severe, Warning, Informational*) rather than on a per-signature basis. Thus it is possible to disconnect session for all *Critical* and *Severe* alerts, report only for *Warning*, and ignore all *Informationa*l alerts. If it is required to alter the action for a particular signature, all that is necessary is to amend its Threat Level. Changing the action for all Informational alerts from *ignore* to *report*, or switching all signatures *en masse* from *disconnect session* to *report only*, is then only a matter of selecting the appropriate action in the Action Settings tab. This is very straightforward and very powerful.

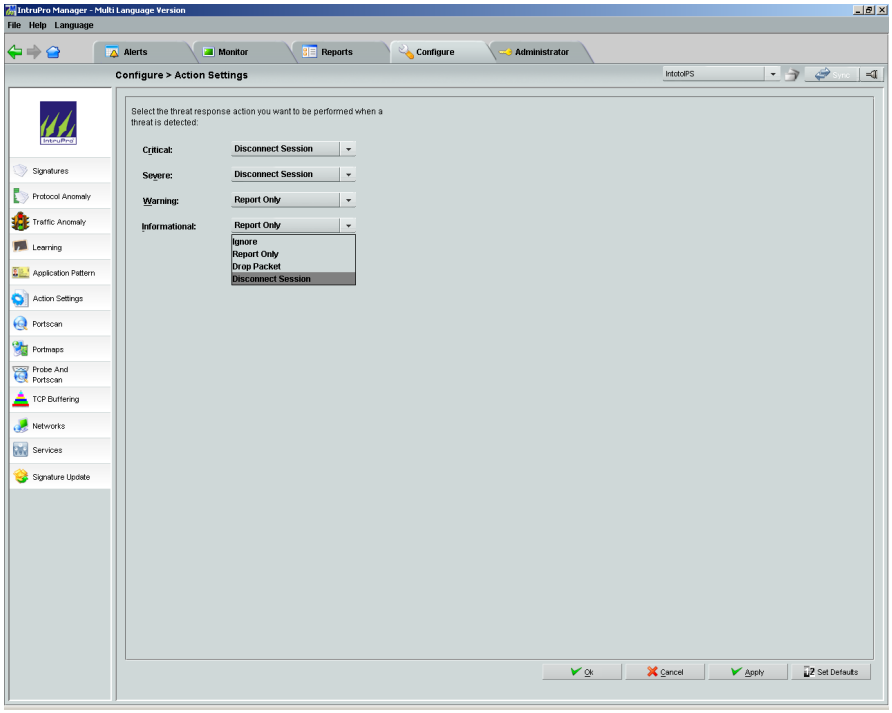

Figure 6 - IntruPro: Action Settings

Of course, it does presume that the signatures have been classified accurately in the first place. For example, we noted some signatures which were in the *Gain Access* or *Exploit* Families but which were classified as *Informational* alerts. This was clearly incorrect, and required that we change the Threat Alert manually on those signatures - however, there is a good chance that such mis-categorisation would go unnoticed by the average user.

This is a clear warning that even (or *especially*) where vendors attempt to make life easy for the administrator, it is still necessary to verify manually that the correct signatures are enabled in the active policy, and that the correct actions are being taken for the appropriate groups of signatures.

Once again, granularity of change is also an issue. It is not possible, for example, to change the Threat Level for all IIS or Apache signatures in a single operation - it would require a search operation, followed by an extended session editing each signature individually. More work is needed in this area.

Other options included in the Configuration tab include:

- ***Protocol Anomaly -*** *There is a list of Protocol Anomaly signatures, and it is possible to view the descriptions, modify certain elements of the signatures individually, and set the traffic monitoring direction for the entire set. It is not, however, possible to enable/disable individual signatures in the section.*
- ***Traffic Anomaly -*** *This allows the administrator to specify certain types of traffic to be rate limited or blocked on a per-connection basis. It is also possible to white-list (allow) and black-list (block) specific services (on specific ports) and URIs. Related reports/graphs include:*
    - ***Connection Graph -*** *This graph shows three lines (open request, established and closed) over a period of time. The statistics table shows total number of open requests, established and closed*
    - ***Application Usage Graph -*** *This graph shows number of bytes transferred per protocol over period of time. The statistics table shows total number of bytes transferred per protocol*
    - ***Network Utilisation Graph -*** *This graph shows number of bytes transferred by a system over a period of time. The statistics table shows total number of bytes transferred per system*
    - ***Traffic Flow Graph -*** *This graphs show number of bytes transferred between two systems over a period of time. The statistics table shows total number of bytes transferred between two systems*
    - ***TCP Reset Graph -*** *This graph shows number of Resets over a period of time. The statistics table shows total number of Resets occurring per protocol*
    - ***Active Connection Graph -*** *This graph shows current active/established connections. The statistics table shows total number of active connections per protocol.*
- ***Learning -*** *Because setting Traffic Anomaly thresholds can be a difficult task, the Learning mode offers the ability to monitor specific connections or types of traffic over a period of time to gain insight into maximum, minimum and average levels of activity.*
- ***Application Pattern -*** *Allows the administrator to define recognition patterns for custom applications*
- ***Port Map -*** *Using this screen, the administrator can assign traffic on a particular port to an application category. Once the traffic on a port is identified as belonging to a particular category, the rules and signatures corresponding to that application category will be applied on that traffic to detect possible vulnerabilities (i.e. running HTTP on port 9000). All the standard services are pre-defined.*
- ***Port Scan -*** *This allows the administrator to define thresholds for port scan and flooding activity.*
- ***Probe and Port Scan -*** *This uses off-line detection capabilities to filter the alert logs to identify possible slow scans, or multiple related scans, after the fact*
- ***TCP Buffering -*** *Attackers could send an attack across multiple small packets to evade the IPS. TCP buffering is designed to detect intrusions that span multiple packets by buffering up to a specific byte delimiter, an entire line, or to buffer up to some value extracted from the first packet*
- ***Networks -*** *Specify internal and external network values (or any)*

- ***Services -*** *Allows the administrator to specify threshold values to be used when detecting protocol anomalies (HTTP header size, maximum URI length, etc.)*
- ***Signature Update -*** *Allows immediate update of the signature library from the Intoto global update server or from a local file. It does not appear as though this process can be scheduled for unattended operation*

Once all changes have been made, the configuration is saved to the Database Store, following which it is synchronised with the Sensor (packet inspection is interrupted briefly during this operation). There is no way to save multiple versions of a configuration nor to roll back to a previous version, and there is no audit trail of configuration changes.

Alert Handling

Rather confusingly, the *Alerts* tab is not the best place to start monitoring alerts. Intoto *Alerts* are actually messages generated by filtering the log files based on user-defined rules, and an Alert is generated when any of its properties are satisfied by the log messages generated by the Sensor.

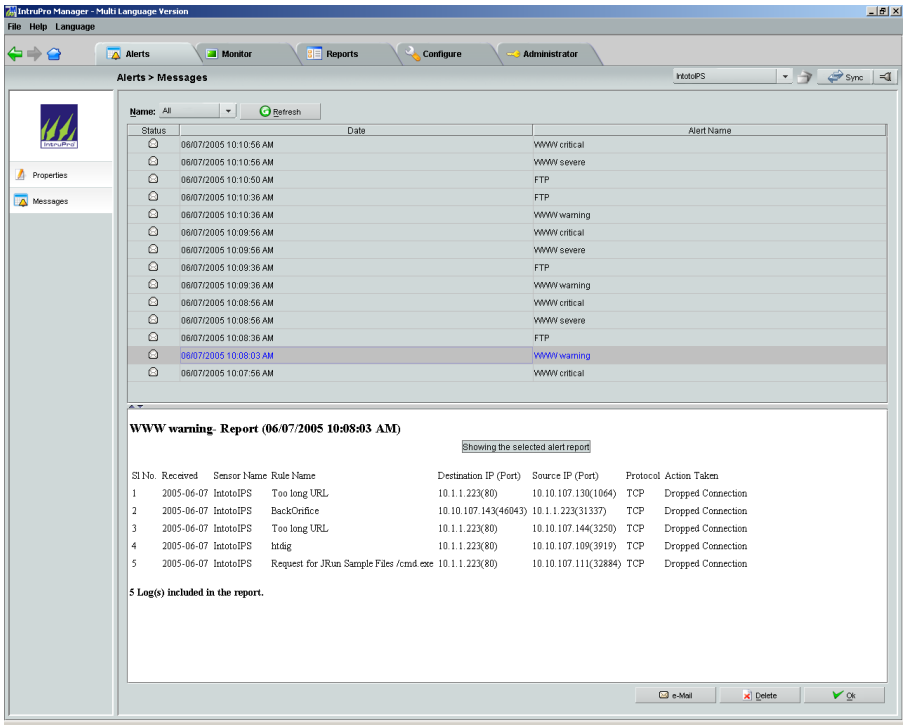

Figure 7 - IntruPro: Alerts

Alert criteria can be defined based on the number of log entries generated within a specified time period, destination IP, source IP, Protocol, Intoto Rule ID, Rule Category (HTTP, SSH, FTP, etc.), and Threat Level (Critical, Informational, etc.). Any number of *Alert Rules* can be defined, and when log entries match one or more Alert Rules, Alerts are generated in the *Alert Messages* window.

Selecting an Alert Message in the upper pane shows the contents in the lower pane, which may consist of a single alert or a number of similar or related alerts (depending on the *time period* and *number of log entries* parameters specified).

Unfortunately, these are fixed lists containing limited alert information, and it is not possible to view more detail or drill down to the relevant log entries.

Alerts are useful as a high-level view of what is being blocked, or for when an administrator wishes to define some very specific criteria for which he wishes to monitor and be informed immediately (i.e. HTTP attacks against a particular Web server). In this respect, they function as a basic correlation capability. More useful, in-depth alert handling is actually provided via the *Reports* tab, however.

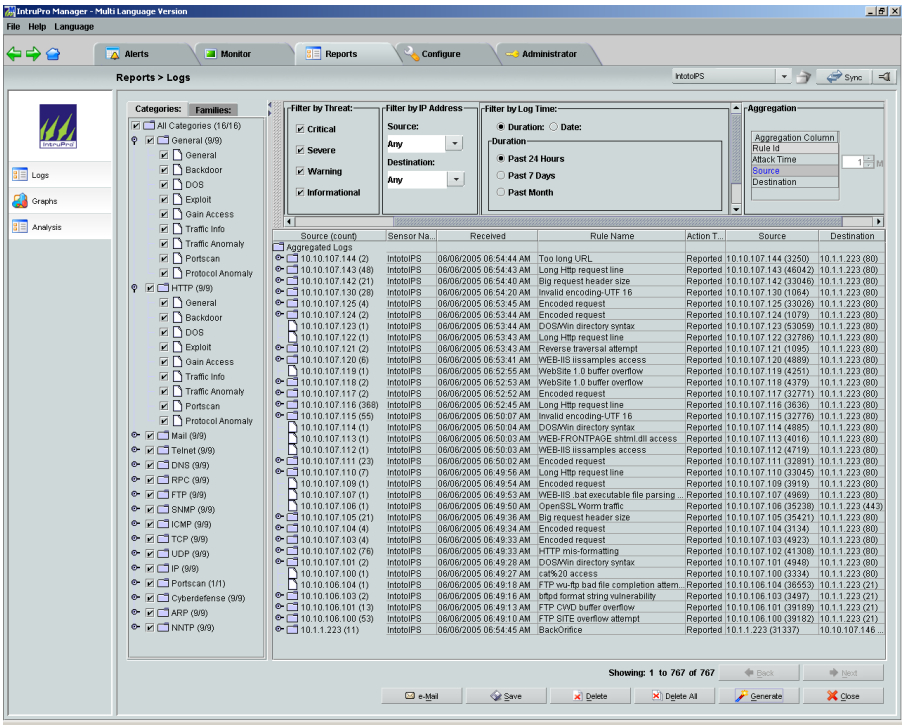

Figure 8 - IntruPro: Reports

The upper pane of the *Reports* screen contains a number of useful selection criteria which are used to determine the log entries displayed in the lower pane:

- ***Category/Family Tree -*** *Allows the administrator to filter out logs by intrusion categories or signature families. Tabs are used to switch between families and categories, and check boxes are used to select entire categories/families or individual signatures via the hierarchical menu structure.*
- ***Sensor Name -*** *Filter by sensor name*
- ***Threat Response -*** *Filter using the options Critical, Severe, Warning and Informational.*
- ***IP Address -*** *Filter logs by Source and Destination drop-down boxes.*
- ***Log Time -*** *Logs can be selected for a specified duration (last hour, last week, last month) or between start/end dates.*
- ***Aggregation -*** *Logs can be sorted and grouped based on the Rule ID, Attack Time, or Source/Destination IP. The Attack Time interval is set in Minutes, determining the time period over which individual log entries will be aggregated.*

Note that it is not possible to filter based on source or destination port, nor on attack description/Rule ID, both of which we consider significant omissions.

We found the Aggregation feature to be particularly useful in reducing the "clutter" caused by single exploits raising multiple alerts, since aggregating based on Source IP ensured that only a single line was displayed on screen for each group of alerts from the same source. When aggregation is active, the summary line shows a count of alerts beneath it, and clicking on a small icon to the left of the summary will expand to show the individual alerts.

Columns can be re-sized and re-ordered by dragging them around the screen, but it is not possible to sort the list based on a specific column, which we found very frustrating. Another frustration was the inability to save filter settings and column layouts for re-use - instead, it is necessary to define a report from scratch every time you wish to run it.

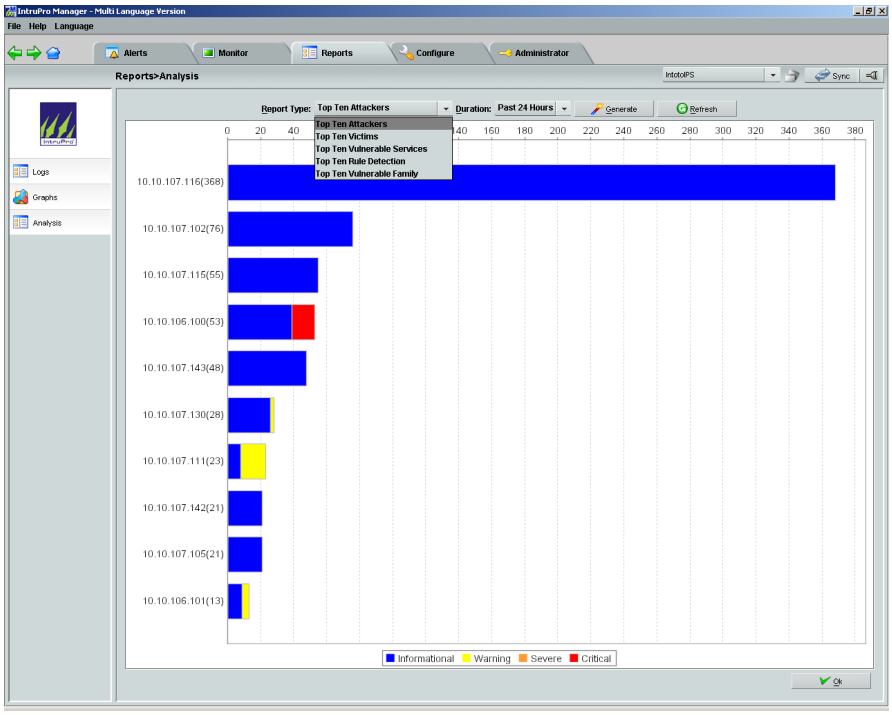

Figure 9 - IntruPro: Graphical reports

Having displayed the alerts based on the selection criteria, the administrator can select all of them or individual alerts and save to an HTML file or e-mail them to a specified address.

Double clicking any line brings up a window showing the detailed alert information:

- *Intoto ID*
- *Attack Time*
- *Log Description*
- *Rule name*
- *Rule Description*
- *Category*
- *Family*
- *CVE ID(s)*
- *BugTraq ID(s)*
- *Scan ID(s)*
- *Threat Level*
- *Protocol*
- *Action Type*
- *Secure network ID*

- *Connection direction (inbound/outbound)*
- *Source IP/Port*
- *Destination IP/Port*
- *IPID*
- *IP Options*
- *TCP Options*
- *TCP Flags*
- *TTL*
- *IP Length*
- *Raw Data/Application Data*

A tremendous amount of detail is provided for each alert, and all of this can be e-mailed or saved to an HTML file at the click of a button. Buttons are also available to allow the administrator immediately to modify, delete or disable the rule, making it very straightforward to tune policies quickly based on alerts raised.

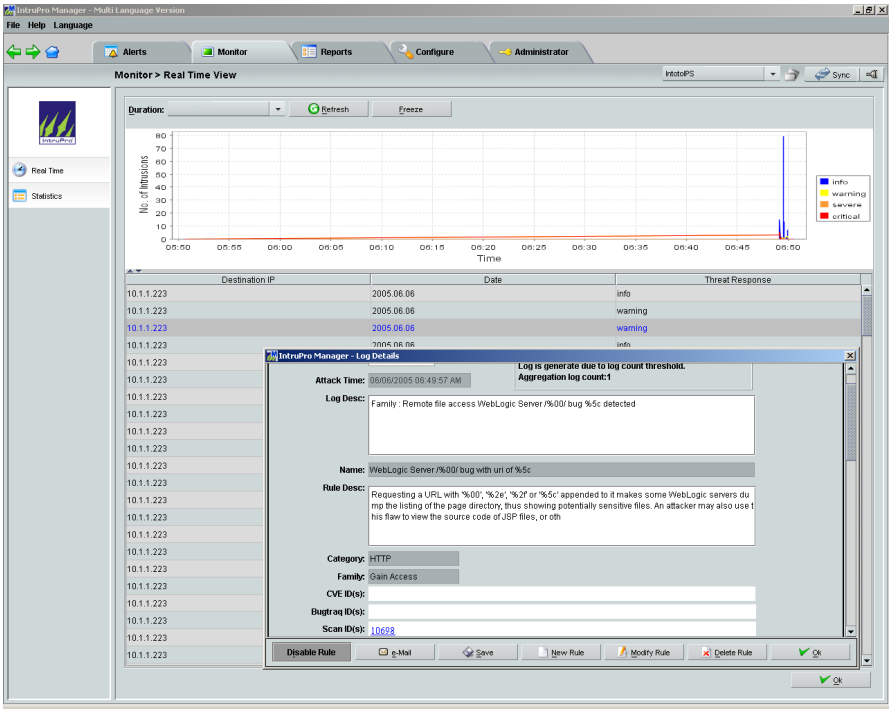

Figure 10 - IntruPro: Real-time Monitor

The Monitor tab provides another real-time view on the current alerts. A graph will display the attacks in a specific time period (selected from a drop-down menu) categorised by threat response.

The lower portion of this screen displays the list of related log messages (limited to the most recent 1000 logs). As with the Alerts screen, these are fixed lists - it is not possible to access further detail on log entries from this screen.

In general, those administrators who are happy to simply know that threats are being blocked will probably be satisfied with the information provided by the *Alerts* and *Monitor* tabs.

Those who want to know a little more detail about what is being blocked, or who want to tune their policy to weed out false positives, will get more use from the *Reports* tab.

Reporting and Analysis

The most easily accessible monitoring screen is the "*Home*" screen which is displayed when the Console is first opened. The *Home* screen displays basic statistics and an overview of the logs collected:

- ***Top Victims -*** *A graph showing the top 10 victims. The IP Addresses are plotted on the Y-axis while the total log entries against each victim are plotted on the X-axis. The administrator can right-click on the graph to save it for future reference.*
- ***Statistics -*** *This comprises statistics collected from the IntruPro Sensor, including: total packets received, total packets processed, total packets dropped, total connection disconnects, last updated time stamp*
- ***Signature Update Information -*** *Displays the Last Updated time stamp information.*

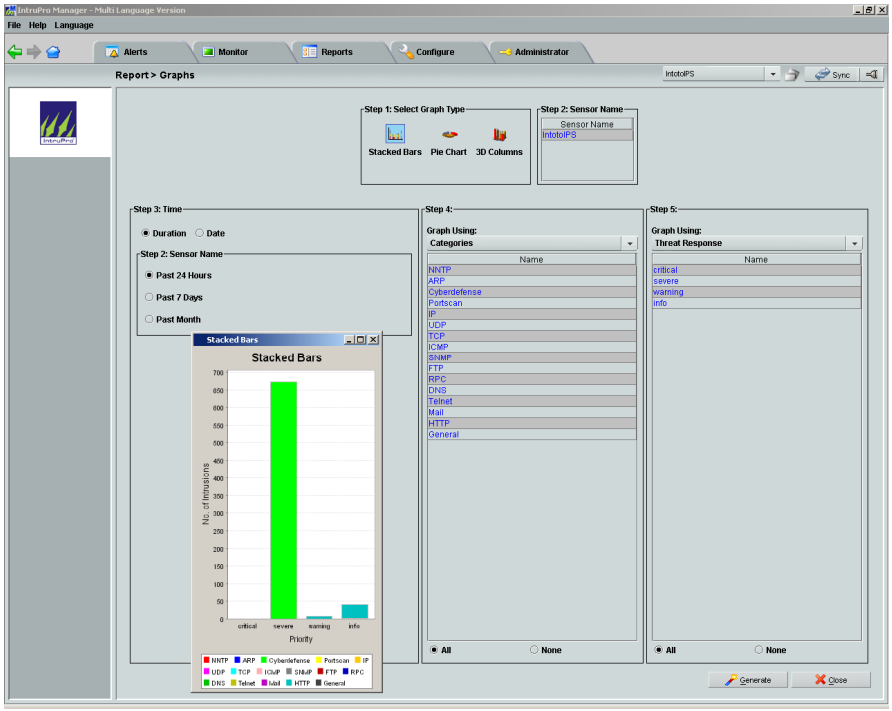

Figure 11 - IntruPro: Analysis screen

The *Graphs* option under the *Reports* tab provides a simple interface to obtain an overall picture of the intrusions detected and protected by the IntruPro Sensor.

A wizard-type interface guides the administrator through defining basic filters to create graphs, and has four steps:

- ***Select Graph Type -*** *Three types of graphs can be generated: Pie Chart, Stacked Bars and 3D Columns.*
- ***Select Sensor Name -*** *Select the required Sensor(s)*
- ***Select Time -*** *Logs can be filtered by pre-defined time periods (last day, week or month) or between two dates.*
- ***Select Group 1 -*** *Group by Categories/Families/Threat Response*
- ***Select Group 2 -*** *Within Group 1, group by Categories/Families/Threat Response*

The resulting graphs can be saved as image files by right clicking on the graph and selecting the option "*Save as*", but they cannot be otherwise printed or exported directly from this interface.

Selecting the *Statistics* option under the *Monitor* tab displays the statistics information of each signature Category (i.e. FTP, HTTP, etc.). Selecting a category from the list in the upper pane displays the total number of intrusions by Threat Category below.

Naturally, the reports tab - which will probably be used to display only those alerts in the last hour or day when being used for alert analysis - can also be used to filter logs over a more lengthy time period for weekly or monthly reports (the same features apply as mentioned in the previous section).

However, given that it is not possible to save report criteria or layout, nor to schedule reports for regular production, we would class reporting as basic in this product.

Verdict

Performance

Note that Intoto IntruPro is a software product and was tested on reference hardware (dual Xeon processor, 2GB RAM, copper Gigabit ports) as a 100Mbps IPS device.

Performance at almost all levels of our load tests was impeccable, with 100 per cent of all attacks being detected and blocked under all load conditions except at the 2000 connections per second level. We would happily confirm Intoto's 100Mbps rating for this device under normal network conditions.

The basic latency figures of IntruPro on the reference hardware were good for a 100Mbps device across the board under all traffic loads, and behaviour throughout the tests with no background traffic was reasonably consistent and predictable, with small decreases as additional network load was applied from 25Mbps to 100Mbps. HTTP response times were also very good.

The device had no problems mitigating our DDOS traffic, although latency did increase significantly during the attack.

IntruPro performed consistently and completely reliably throughout our tests, and exposing the sensor interface to ISIC-generated traffic had no adverse effect on the device at any time.

Security Effectiveness

Out of the box, blocking performance was good at 80 per cent, and was improved to 95 per cent following the application of a signature update after 48 hours. Detection/recognition rate was slightly lower (89 per cent following update) due to the fact that many DOS and ICMP attacks are blocked without alerting by the firewall module.

We also noted that many of our test cases raised several alerts for a single exploit. This can be annoying, and we would prefer that the product performed simple correlation to reduce the number of superfluous alerts raised.

It can, however, be mitigated by the fact that the source aggregation feature of the management interface results in a single entry on the GUI which can then be expanded to show the individual alerts if required.

Performance in our “false negative” tests was good out of the box, and improved following the signature update, leaving just a single miss out of 14.

The device initially failed five test cases in our false positive test suite, and although all of these were eliminated following the signature update, we also noted several false positives during our FTP testing, the result of some sloppy port-based backdoor signatures. Overall, false positives would be a concern for us with this device at present, and we would encourage potential purchasers to test in their own network before buying.

Resistance to known evasion techniques was excellent, with IntruPro achieving a clean sweep across the board in almost all of our evasion tests. *Fragroute* and *Whisker* both failed to deceive the device into ignoring valid attacks, and all but one of the attempts were decoded accurately. Some minor configuration was required to enforce blocking for two of the test cases, but this should become the default configuration in future releases. Most of the miscellaneous evasion techniques (including RPC record fragging) were also handled well.

During testing, we determined that the IntruPro device was able to handle up to 45,000 simultaneously open connections whilst successfully maintaining state on our half-open exploits. We consider this to be just adequate for a 100Mbps device.

Stateless “exploits” are blocked by default, with no alerts. This behaviour is not configurable, and we would prefer to see a grace period following device start-up during which existing connections (which will appear as mid-flows initially) are passed successfully.

Usability

The IntruPro Manager provides a usable and straightforward environment for configuring and managing a single Sensor, or small number of Sensors. However, there is no role-based user access and no centralised policy management/distribution capabilities which limits the scope of this product to smaller deployments.

Initial configuration is made extremely simple via the ability to click on a single button to select all recommended signatures, and then apply the required actions *en masse* via the *Action Settings* tab. This way, it is very simple to switch between a “monitor only” and “block traffic” mode of operation, even though there is no specific “pass through” capability.

The search facility within the Signature tab makes it easy to extract a number of related signatures for editing, though it would be nice to have some form of bulk editing capability rather than have to then edit them individually.

There are a number of different ways to view alerts, including a very useful custom *Alerts* tab whereby the administrator can define criteria to match against the logs. This equates to a basic form of high-level correlation, ensuring that only the most relevant log entries are brought to his attention, and can be extremely useful. Also of use are the summary graphs provided in the *Monitor* tab.

Simple, high-level views in the *Alerts* and *Monitor* sections can thus be used to provide a rapid indication of what is being blocked, whilst those who wish to pursue a more detailed investigation will find the *Reports* tab offers most of what they require.

The Reports section provides in-depth analysis via custom filters, though it is missing a couple of key filtering options (port and signature ID, for example). In addition, the administrator cannot sort on columns in Reports screen (a standard Windows feature) and it is not possible to save report layouts and criteria for re-use. This necessitates that every report is set up from scratch every time it is run.

It is unfortunate that there is no direct drill-down capability from the Alerts or Monitor displays into the detailed log views, but double-clicking on any alert allows it to be edited or disabled directly from the Report screen - very useful for tuning Sensor configuration. An aggregation function provides multiple means to sort and group alerts to reduce on-screen clutter.

In the present version, we would class both reporting and alert handling as basic, though adequate for the task in hand. We also feel that although the current system is adequate for single-Sensor management tasks, it is not scalable.

Contact Details

Company: Intoto Inc.

E-mail: info@intoto.com or sales@intoto.com

Internet: http://www.intoto.com

Address:
3100 Dela Cruz Blvd #300
Santa Clara
CA 95054
USA

Tel: +1 408 844 0480
Fax: +1 408 844 0488

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.*

Intoto IntruPro V3.0

Section 1 - Detection Engine

Test 1.1 – Attack Recognition	Attacks	Default ARRD	Default ARRB	Custom ARRD	Custom ARRB
Test 1.1.1 - Backdoors	7	5	5	7	7
Test 1.1.2 - WINS/DNS	3	1	1	2	2
Test 1.1.3 - DOS	10	6	10	6	10
Test 1.1.4 - False negatives (modified exploits)	14	11	11	13	13
Test 1.1.5 - Finger	4	4	4	4	4
Test 1.1.6 - FTP	5	4	4	5	5
Test 1.1.7 - HTTP	43	34	34	40	40
Test 1.1.8 - ICMP	2	0	2	0	2
Test 1.1.9 - Reconnaissance	8	8	8	8	8
Test 1.1.10 - RPC	9	6	6	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	0	0
Total	**110**	**83 / 110**	**89 / 110**	**98 / 110**	**104 / 110**
		75%	**81%**	**89%**	**95%**

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	PASS	PASS
Test 1.2.6 - Normal NetBIOS traffic	PASS	PASS
Test 1.2.7 - POP3 e-mail containing “suspicious” URLs	FAIL	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	FAIL	PASS
Test 1.2.10 - SMTP e-mail transfer containing “suspicious” URLs	FAIL	PASS
Test 1.2.11 - SMTP e-mail transfer with “suspicious” DLL attachment	FAIL	PASS
Test 1.2.12 - SMTP e-mail transfer with “suspicious” Web page attachment	FAIL	PASS
Test 1.2.13 - SNMP V3 packet with invalid parameter	PASS	PASS
Test 1.2.14 - Fake DNS /bin/sh buffer overflow	PASS	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	**12 / 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	YES	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[1]
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**	**20 / 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	NO	YES[1]
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**	**8 / 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	YES	YES	YES
Test 2.4.3 - Inserting non-text Telnet opcodes in FTP data stream	YES	NO[3]	YES
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[2]	YES[2]	YES[2]
Total	**6 / 7**	**5 / 7**	**6 / 7**

Section 3 - Stateful Operation

Test 3.1 – Stateless Attack Replay	Alert?	Blocked?	Pass/Fail
Test 3.1.1 - Stateless Web exploits	NO[4]	YES[4]	PASS
Test 3.1.2 - Stateless FTP exploits	NO[4]	YES[4]	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	N/A[5]	N/A[5]	N/A[5]	N/A[5]
Test 3.2.2 - Attack Blocking	PASS	PASS	PASS	N/A[5]	N/A[5]	N/A[5]	N/A[5]
Test 3.2.3 - State Preservation	PASS	PASS	PASS	N/A[5]	N/A[5]	N/A[5]	N/A[5]
Test 3.2.4 - Legitimate traffic blocking	PASS	PASS	FAIL[5]	N/A[5]	N/A[5]	N/A[5]	N/A[5]

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	N/A[5]	N/A[5]	N/A[5]	N/A[5]
Test 3.3.2 - Attack Blocking	PASS	PASS	PASS	N/A[5]	N/A[5]	N/A[5]	N/A[5]
Test 3.3.3 - State Preservation	PASS	PASS	PASS	N/A[5]	N/A[5]	N/A[5]	N/A[5]
Test 3.3.4 - Legitimate traffic blocking	PASS	PASS	FAIL[5]	N/A[5]	N/A[5]	N/A[5]	N/A[5]

Section 4 - Detection/Blocking Performance Under Load

Test 4.1 – UDP traffic to random valid ports		25Mbps	50Mbps	75Mbps	100Mbps	Max
Test 4.1.1 - 256 byte packet test - max 45,500pps	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	
Test 4.1.2 - 550 byte packet test - max 22,000pps	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	
Test 4.1.3 - 1514 byte packet test - max 12,000pps	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	

Test 4.2 – HTTP "maximum stress" traffic with no transaction delays		25Mbps	50Mbps	75Mbps	100Mbps	Max
Test 4.2.1 - Max 250 connections per second - ave packet size 1000 bytes - max 12,000 packets per second	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	
Test 4.2.2 - Max 500 connections per second - ave packet size 540 bytes - max 22,500 packets per second	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	
Test 4.2.3 - Max 1000 connections per second - ave packet size 440 bytes - max 27,500 packets per second	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	
Test 4.2.4 - Max 2000 connections per second - ave packet size 360 bytes - max 32,000 packets per second	Detected	100%	100%	100%[6]	N/A[6]	70Mbps
	Blocked	100%	100%	100%[6]	N/A[6]	

Test 4.3 – HTTP "maximum stress" traffic with transaction delays		25Mbps	50Mbps	75Mbps	100bps	Max
Test 4.3.1 - Max 500 connections per second - ave packet size 540 bytes - max 22,500 packets per second - 10 sec delay - max 5,000 open connections	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	
Test 4.3.2 - Max 1000 connections per second - ave packet size 440 bytes - max 27,500 packets per second - 10 sec delay - max 10,000 open connections	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	

Test 4.4 – Protocol mix		25Mbps	50Mbps	75Mbps	100Mbps	Max
Test 4.4.1 - 72% HTTP (540 byte packets) + 20% FTP + 6% UDP (256 byte packets). Max 400 connections per second - ave packet size 540 bytes - max 22,000 packets per second - max 75 open connections	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	

Test 4.5 – Real World traffic		25Mbps	50Mbps	75Mbps	100Mbps	Max
Test 4.5.1 - Pure HTTP (simulated browsing session on NSS Web site). Max 475 connections per second - 4 new users per second - ave packet size 560 bytes - max 21,000 packets per second	Detected	100%	100%	100%	100%	100Mbps
	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 375 connections per second - ave packet size 560 bytes - max 21,000 packets per second - max 150 open connections	Detected	100%	100%	100%	100%	100Mbps
	Blocked	100%	100%	100%	100%	

Section 5 - Latency & User Response Times

Test 5.1 – Latency	Packet Size	25Mbps	50Mbps	75Mbps	100Mbps
Test 5.1.1 Average latency (µs) with no background traffic	256	157.16	157.35	153.14	156.39
	550	214.52	182.75	182.49	185.12
	1000	254.98	247.73	223.04	222.02
Test 5.1.2 Average latency (µs) with background traffic (50Mbps HTTP traffic, max 250 connections per second - ave packet size 540 bytes - max 12,000 packets per second)	256	220.57			
	550	243.72			
	1000	289.53			
Test 5.1.3 Average latency (µs) when under attack (10Mbps SYN flood (14800pps))	256	15590.21[7]			
	550	38234.63[7]			
	1000	32074.35[7]			

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 (50Mbps HTTP traffic, max 250 connections per sec - ave packet size 540 bytes - max 12,000 packets per sec)	172632	0	202	359	203
Test 5.2.2 - Web page response (ms) when under attack (10Mbps HTTP traffic, max 250 connections per sec - ave packet size 540 bytes - max 12,000 packets per sec PLUS 10Mbps SYN flood (14800pps))	171637	3	202	29738	623

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?	YES

Notes:

1. Does not block by default - requires simple configuration. This will be rectified in future releases.
2. HTTP exploits can be detected on any port with configuration
3. Invalid control characters are detected, but the original exploit is not decoded
4. This behaviour is not configurable. There is no “grace period” to allow mid-flow connections on Sensor start-up
5. Maximum simultaneous open connections accomplished under test was approx 45,000
6. Maximum achieved under test was 1400 connections per second (at 70Mbps)
7. Some lost packets during SYN flood caused high average latency figures

Section 1: Detection Engine

We installed one sensor with the latest signature library, and enabled **all** signatures, with traffic monitoring in both directions.

Out of the box, blocking performance was good at 80 per cent, and was improved to 95 per cent following the application of a signature update after 48 hours.

Detection/recognition rate was slightly lower (89 per cent following update) due to the fact that many DOS and ICMP attacks are blocked without alerting by the firewall module. We also noted that many of our test cases raised several alerts for a single exploit. This can be annoying, and we would prefer that the product performed simple correlation to reduce the number of superfluous alerts raised. It can, however, be mitigated by the fact that the source aggregation feature of the management interface results in a single entry on the GUI which can then be expanded to show the individual alerts if required.

Performance in our “false negative” tests was good out of the box, and improved following the signature update, leaving just a single miss out of 14.

A major concern in deploying an IPS is the blocking of legitimate traffic. The device initially failed five test cases in our false positive test suite, and although all of these were eliminated following the signature update, we also noted several false positives during our FTP testing, the result of some sloppy port-based backdoor signatures. Overall, false positives would be a concern for us with this device at present, and we would encourage potential purchasers to test in their own network before buying.

Section 2: IPS Evasion

Resistance to known evasion techniques was excellent, with IntruPro achieving a clean sweep across the board in almost all of our evasion tests. *Fragroute* and *Whisker* both failed to deceive the device into ignoring valid attacks, and all but one of the attempts were decoded accurately. Some minor configuration was required to enforce blocking for two of the test cases, but this should become the default configuration in future releases.

Of the miscellaneous evasion techniques, only changing ports on backdoors caused problems (a not untypical situation, since it is expensive on resources to monitor for all backdoors on all ports), but the rest (including extensive RPC record fragging) were handled well.

Section 3: Stateful Operation

Out of the box, Intoto claims that the IntruPro on 100Mbps reference hardware can handle up to 50,000 open connections. This was not quite confirmed in testing, the device reaching its limits at around 45,000 open connections. We consider this to be just adequate for a 100Mbps device.

During testing, we verified IntruPro’s ability to handle up to 45,000 simultaneously open connections whilst successfully maintaining state on our half-open exploits.

The default operation of the device is to reject new connections when the state tables are full or resources are low, and this means that once the connection limit is exceeded, the device continues to maintain state on existing connections (thus detecting our half-open exploits), but inevitably, as a consequence, begins to block legitimate traffic. This behaviour is not configurable at run time.

Stateless “exploits” are blocked by default, with no alerts. This behaviour is not configurable at run time, and there is no grace period following device start-up during which existing connections (which will appear as mid-flows initially) will be passed successfully.

Section 4: Detection/Blocking Performance Under Load

Note that Intoto IntruPro is a software product and was tested on reference hardware (dual Xeon processor, 2GB RAM, copper Gigabit ports) as a 100Mbps IPS device.

Performance at almost all levels of our load tests was impeccable, with 100 per cent of all attacks being detected and blocked under all load conditions except at the 2000 connections per second level. In *Test 4.2.4* we noted that the maximum connection rate was around 1400cps, which equates to 70Mbps.

We would happily confirm Intoto’s 100Mbps rating for this device under normal network conditions, however.

Section 5: Latency & User Response Times

The basic latency figures of IntruPro on the reference hardware were good for a 100Mbps device across the board under all traffic loads. They ranged from 157µs with 25Mbps of 256 byte packets, to 222µs with 100Mbps of 1000 byte packets.

Behaviour throughout the tests with no background traffic was reasonably consistent and predictable, with small decreases as additional network load was applied from 25Mbps to 100Mbps. Placing the device under a half load of 50Mbps of HTTP traffic increased latency somewhat, rising from 157µs to 220µs with 256 byte packets, from 214µs to 243µs with 550 byte packets, and from 254µs to 289µs with 1000 byte packets.

HTTP response times were also very good, demonstrating an average of 203ms with 50Mbps of HTTP traffic.

Latency when under 10Mbps (14800 packets per second) of SYN flood traffic was excessive due to packet loss. Despite this, the SYN flood mitigation technique (SYN proxy) used by Intoto was very effective, with very little of the DOS traffic making its way through to the target server. Although average HTTP response time for legitimate traffic increased to 623ms during the attack, there were very few failed transactions,.

It is worth noting that the worst performance in terms of both latency and packet loss was seen when the SYN flood was directed against port 80 - other ports were handled better. This could indicate a problem with the HTTP detection engine (possibly related to the low connections per second limit) rather than a general inability to handle flood traffic.

Section 6: Stability & Reliability

IntruPro 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 on the device at any time, and relevant protocol anomaly alerts were raised. It is worth noting that the ISIC flood (against multiple ports) was handled much better than the SYN flood attack against port 80.

All other malicious traffic continued to be blocked successfully during the attack, and there were no residual stability problems once the ISIC attack had been terminated.

Section 7: Management Interface

No ports were visible to port scanners.

The extended ISIC attack against the management interface caused communications to be interrupted between the console and the management server/sensor, but these were restored successfully once the attack was terminated and there were no residual stability problems.

Alerts were raised as a result of the attack against the management interface, and all alerts were stored on the Sensor and transmitted to the management server once the ISIC attack had been terminated.