

# False Positives: A User's Guide to Making Sense of IDS Alarms

*by Marcus J. Ranum for ICSA Labs IDSC  
February 2003*

## Participating Vendors



---

# False Positives: a User's Guide to Making Sense of IDS Alarms

*by Marcus J. Ranum for ICSA Labs IDSC*

---

## Table of Contents

Executive Summary. . . . .	3
Definitions. . . . .	4-5
False Positives vs. Noise. . . . .	6-7
Location. Location. Location. . . . .	8-9
Processing in the Console and Back-End Systems. . . . .	10
Measuring IDS Effectiveness. . . . .	10-12
Summary. . . . .	12

# Executive Summary

Intrusion Detection Systems (IDS) became widely available as commercial off-the-shelf (COTS) security products starting in the late 1990's. Since that time, they have demonstrated their worth, but not their full potential, by assisting network administrators in detecting holes as they are being exploited, or by providing valuable forensic data during clean-up after networks have been penetrated by attackers. It is safe to say that IDS have been a mixed blessing: they provide both priceless data and time-wasting irritation – sometimes in nearly equal measure.

Some industry pundits have declared the technology “not ready for prime-time” while others have suggested CIOs and CTOs add it to their “must-have” list. Why? Because dealing with the myriad alarms from IDS can be expensive in terms of staff-time and requires a great deal of expertise on the part of the security or network administrator. Worse, the results are highly dependent on each individual network and its traffic profile – it is very hard to generalize results from one network to another. Many organizations are blissfully unaware of how their network is being used; installing an IDS is usually a critical first step in learning how to gain control of the network.

In this paper, we'll look at the types of alarms that are returned by IDS and their relative value. We will begin by offering some definitions of IDS alert terminology, and we'll conclude by outlining some ways you can assess the value of the information that is returned from your individual network. In between, we will examine some of the issues that crop up when testing IDS: how do you determine what tests are even worth throwing against an IDS and what should you expect them to return? Bearing in mind that IDS is a technology that is evolving rapidly we will try to look not only at features of the current crop of IDS products, but features that may not yet be widely available.

# Definitions

- **Alert / Alarm (Or "True Alarm")** – An alarm that identifies a system that has just been successfully attacked. Typically, alarms also include diagnostic information regarding the context in which the attack took place but it is possible for them to simply be alarms regarding anomalous or statistical events. The terms "Alert" and "Alarm" are interchangeable in IDS nomenclature.
- **False Positive** – An alarm generated by an IDS in which the IDS alerts to a condition that is actually benign. In other words, **the IDS made a mistake**. A typical example of a false positive would be a case when an IDS raises a "SYN flood" alarm because it sees a large number of SYN packets directed at a busy web server and mistakenly concludes it is under attack. Another example of a false positive would be an IDS raising a "SMTP Wiz attack" alarm when it observes the string "DEBUG" in the body of an SMTP message.
- **False Negative** – A non-event in which an IDS fails to generate an alarm when an alert-worthy condition is in effect. A typical example of a false negative might be an IDS' failure to detect a web-server buffer directory traversal attack because the attacker developed a previously unknown way of obscuring the filename that is being requested. Another example might be a false negative resulting from an IDS' failure to capture all the packets necessary to accurately reassemble an attacker's action due either to network load or changes in routing topology.
- **Noise** – An alarm in which the IDS sends an alert on a condition that is non-threatening or not applicable to the site that is being monitored, but which is correctly diagnosed. **The IDS did not make a mistake** but the alarm is of questionable value. For example, the IDS might identify a Windows/Intel-based buffer overrun against a Solaris/SPARC system that will not be affected by the attack. Another example of noise might be an alarm in which an IDS outside a firewall correctly identifies hostile scanning activity that the site administrator knows will not penetrate past the firewall. In the first example, the IDS correctly identified an event of possible interest – site administrators will most likely react with varying levels of concern depending on the origin, persistence, or target of the attack. In the second example, since the IDS is outside the firewall, a certain amount of hostile traffic is expected, even if it is ignored or deemed insignificant. Some sites employ IDS specifically to collect statistics regarding attacks launched against their outer perimeter – the "noise level".
- **False Attack Stimulus** – A stimulus that causes an IDS to trigger an alarm when no actual exploited attack has occurred. False attack stimuli generate false positives; and are frequently seen during badly designed IDS tests or when attackers attempt to overload an IDS' alert processing capability using a tool such as Stick. Many scanning tools generate false attack stimuli. For example, if a vulnerability assessment tool connects to a web server and issues a "GET" for a known-vulnerable CGI-bin script, it is not the same thing as when a hacking tool connects and exercises the complete attack via the same script.

Depending on the application protocols in use it may be difficult for the IDS to distinguish a stimulus that looks for a vulnerability from a stimulus that actually triggers a compromise in the system. False attack stimuli are deliberately used in some IDS testing regimens, attempting to verify the IDS' function without placing real systems at risk. When testing IDS a tester should mix a number of false attack stimuli with true attack stimuli.

- **True Attack Stimulus** – A stimulus that is indistinguishable from an actual successful attack. For example, in a TCP-based network attack, a True Attack Stimulus would consist of a valid TCP session set-up with 3-way handshake, the attack payload, and a TCP session shutdown. In order to produce True Attack Stimuli, it is usually necessary to either actually attack vulnerable systems, or to simulate the attack using session replays of actual attacks on vulnerable systems.

- **Site Policy Awareness** – Sometimes referred to as “Smart IDS”, an IDS’ ability to rewrite its alarm values based on site policy or knowledge of site configuration. For example, an IDS might be configured to apply different priorities to alarms based on the administrator’s specifying the importance of various systems, subnets, or users on the network. Different priorities might be attached to systems identified as mission critical or certain user accounts having been identified as privileged or important.
- **Confidence Value** – A confidence value may be included with an alarm to indicate the IDS’ approximate certainty that it has correctly identified and detected an attack. For example, an IDS might generate an alarm indicating a 90% confidence that a particular attack is a SYN flood based on the ratio of SYN packets to SYN+ACK packets seen coming from a web server. Confidence values are usually assigned by the author of the signatures or detection algorithms generating the alarm, but may be adjusted based on site knowledge. Administrators might wish to filter IDS alarms based on the confidence value attached by the IDS.
- **Alarm Filtering** – Allowing an administrator to select specific alarms to discard based on site configuration. This capability is widely offered in most IDS products today, and is often the primary mechanism for “tuning false positives”: deciding which “false” alarms to throw away. Alarms are generally filtered based on: alarm type, alarm severity, confidence value, target IP address, source IP address, target operating system type or a combination thereof.
- **Alarm Rewriting** – Allowing an administrator to selectively modify alarms as they are produced or received based on site configuration. For example, an administrator might configure the IDS to rewrite and raise the priority of alerts sent regarding a mission critical system. Or, perhaps the priority of alerts regarding Windows-oriented attacks against known UNIX systems might be rewritten to zero. Alarm Rewriting is a very similar functionality to Alarm Filtering, since basically Alarm Filtering consists of rewriting an alarm into having a priority that makes it invisible to the administrator.
- **Alarm Normalization** – Converting a variety of alarm types from various devices into a normalized “internal” form for correlation, clustering, or compaction. Typically, Alarm Normalization is a necessary first step towards importing alarms from heterogeneous systems into a database.
- **Alarm Clustering** – Taking multiple copies of nearly identical alarms and clustering them to create a single high-level alarm. During this process information about the ancestors is typically not discarded. For example a new descendant alarm record might be created within the system at a different priority level than its ancestors, which might be marked as hidden unless specifically requested, or otherwise represented as belonging with the descendant record. Alarm Clustering and Alarm Correlation (see below) are closely related.
- **Alarm Compaction** – Clustering alarms based on frequency, common values, or higher-level value. For example, instead of recording one alarm for each time a system is scanned by the CodeRed worm, a single descendant alarm would be recorded saying “**X** CodeRed scans detected within the last **Y** time.” An example of compaction based on a higher-level value would be compacting all of the “ignore” significance alarms that occurred within a certain time window into a descendant alarm saying “**N** ignore-priority alarms occurred during the last **M** minutes.”
- **Alarm Correlation** – Clustering alarms based on either statistical measures of co-relevance or on knowledge-based matching rules. This capability is usually thought of in the context of correlating alarms from multi-vendor IDS or correlating IDS alarms with firewall or host log data.

*Note: These definitions are intended as a departure point for terminology in the future. As IDS technology continues to evolve, vendors will innovate and attach new terms to their innovations. To “future-proof” these definitions, we’ve deliberately kept them as product-neutral and free of “marketing speak” as possible. Other than defining alarm clustering, compaction, correlation, rewriting, and filtering, we will not spend further time on alarm processing operationsexcept for where alarm processing is used to alter the visibility of noise and false positives.*

# False Positives vs. Noise

Perhaps the biggest misunderstanding about false positives is their relationship with noise. Often, when network managers complain about the false positives from their IDS, they are, in fact, functioning properly and are generating noise. Noise usually results from several causes:

- Placement of the IDS outside of the security perimeter – the IDS is situated in a place where it sees attack traffic as a matter of course. Generally, administrators who place an IDS outside of their network perimeter should expect between dozens and thousands of noise alerts per day, depending on the state of the Internet at a given time. During the CodeRed outbreak, the author saw an IDS generate over 100,000 alerts before it overloaded its backend data systems and began to discard event records.
- Site policy that allows an activity that causes IDS alarms – some networks authorize activities that other network administrators and IDS designers believe is suspicious. In this case, the IDS designers usually err on the side of caution and generate an alert. After all, that's what the IDS is supposed to do. One site the author is familiar with permits outgoing Nmap scanning from individual users; their IDS generated a tremendous number of noise alerts until that IDS signature was turned off.
- Lack of site awareness in the IDS – many IDS lack the ability to determine if a targeted system is vulnerable to the attack launched against it. Or, even if they are able to determine vulnerability, the attack is still detected correctly, and an alert is raised. This is one area where customer expectations confound the IDS designer: does the customer want to know about *all attacks* or just *successful attacks*? What many vendors and customers refer to as “tuning” an IDS is primarily the process of manually adding site awareness to the IDS' policies.

A lot of IDS customers treat noise alerts as if they were an annoyance. Unfortunately, they're usually a symptom of the IDS working correctly but requiring tuning. The usual vendor's response to the types of noise described above is to turn off those signatures, or to re-prioritize the alerts to a lower significance. In the case of an IDS that is deployed outside of a firewall, turning the signatures off usually has less value than re-prioritizing the alerts – after all, if the customer didn't want noise, they wouldn't have installed their IDS outside of their firewall.

Many false positives are recorded during real-world deployments when the customer is running custom applications or unusual software configurations. Indeed, some normal software configurations can confuse an IDS into making mistakes. For example, some types of load-balancing devices send probe packets to backend systems to test their uptime or responsiveness.

These probes may appear to an IDS to be attack reconnaissance probes and generate an alarm. In this situation, the best solution is to disable the signature check for traffic from the load-balancer, since the information gathered is not ever likely to be useful. Extremely busy web sites with large numbers of users on dial-up connections often receive bursts of SYN packets or retransmitted SYN packets – sometimes resulting in a “SYN Flood denial of service” alert from the IDS. Some IDS administrators might choose to turn off that signature to end the false positives, other administrators might leave it turned on with a lowered priority as a means of detecting unusual loading situations in the web site.

The worst case scenario with a false positive is when a common activity on the network consistently triggers a high priority alarm. In such a circumstance, the customer is advised to work with the IDS vendor to identify why their system is tripping an incorrect alarm.

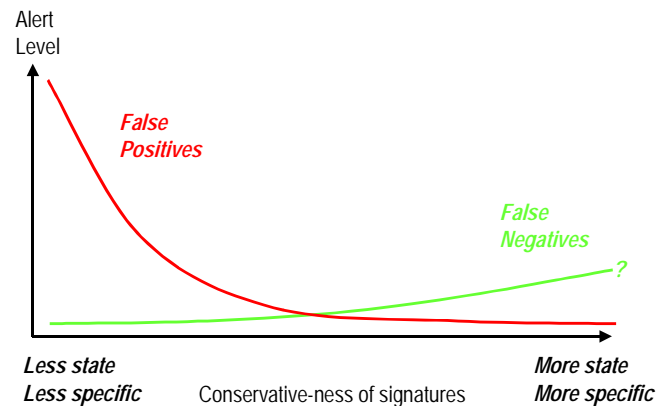
- Turning a signature off because of false positives is a short-term fix while waiting for the vendor to improve the signature
- Turning a signature off because of site configuration, policy, or noise level is a way of applying site-specific knowledge to tune the IDS – it doesn't represent a flaw in the IDS product

In the early days of the IDS market, one vendor was invited to a “shoot-out” of IDS products at a trade show. The organizers of the event supplied the IDS vendor with a partial list of the attacks that were going to be launched within the test network. One of the tests was a particular attack launched against Sun’s NFS remote filesystem protocol; no known IDS product at the time was capable of performing the necessary protocol decoding to analyze the contents of NFS packets.

Prior to the start of the contest, one of the vendors configured their IDS to generate an alert if it saw *any* NFS traffic on the target network. In the case of the limited test, this was a valid signature but it obviously would generate too many false positives on a production network with real (non-hostile) NFS traffic. This simple deceit allowed the vendor in question to claim top honors.

Fundamentally, all IDS embody a trade-off between being too sensitive and annoying their users and being too narrow-focused and missing an important event. **Figure 1** illustrates the *estimated* trade-off based on the author’s practical experience.<sup>1</sup> Note that the rate at which false positives are generated drops off fairly sharply as the signature set adds state or error checking and becomes more precise. The rate of false negatives is not believed to be symmetrical but is actually unknown since the number of unknown attacks is an unknown. However, a signature that does protocol correctness checking to any significant degree dramatically increases the likelihood of catching a large number of unknown attacks – while a single correction to a signature may only remove a single false positive.

**Figure 1:**  
Relationship of signature quality to IDS responsiveness



When an IDS designer produces a signature set and detection logic in a product, it embodies their assessment of how their end-users adjudge the relative value of false positives, noise, and false negatives. In today’s IDS product market, the designers’ ability to judge these values has been somewhat hampered by the way in which customers and trade journal reviewers have been performing IDS tests.

The typical IDS test consists of running a baseline traffic mix and injecting attacks into it. The IDS is then scored based on how many attacks it saw; attacks missed are psychologically weighted more heavily in readers minds because, after all, *if the IDS can’t detect a known attack, it can’t be very accurate, can it?*

So goes the logic, anyhow. The end result is that most vendors will prefer to produce a few more false positives than risk a poor showing in a product review. The noise level an IDS produces has less to do with the quality of its signature set and more to do with the environment in which it is deployed and its ability to capture site-specific knowledge.

<sup>1</sup> In other words, the chart is not based on a measurable assessment; it is intended as an illustration, not science.

# Location. Location. Location.

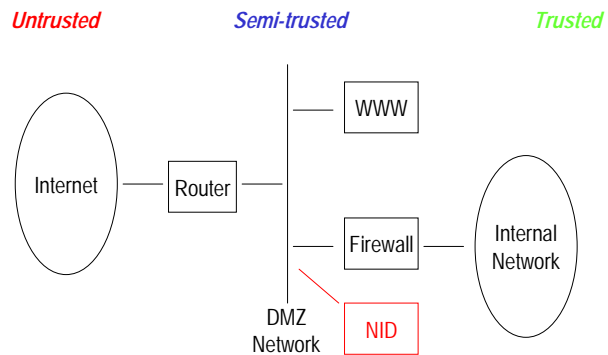
Other than its signature set and analysis engine, the location of the IDS is the factor that most closely governs the number of alerts it generates. The differences between Network IDS (NIDS), Host IDS (HIDS) and log-based approaches are so profound that it's nearly impossible to compare them effectively.<sup>2</sup> For the sake of illustrating an important point, however, we are going to briefly focus some aspects of how NIDS function when in a common network with a firewall. There is considerably more complexity than meets the eye, and it can affect how the IDS reacts in terms of noise or false positives.

In the configuration illustrated in **Figure 2**, we'd expect have a fair amount of noise, since the NID will be detecting attacks launched at the outward-facing WWW server – for the WWW server to be accessible to the outside world, HTTP traffic must reach it on port 80 TCP. Of course, port 80 TCP is a popular port for reconnaissance probes and attacks. CodeRed, for example, scans networks searching for WWW servers port 80. We'd expect the NID to generate a fairly large amount of noise shortly after it went operational.

So, typically, we'd do one of two things:

- Filter some traffic at the router
- Tune the IDS filter set to reduce the noise level

**Figure 2:**  
Sample network with firewall/routers and NIDS



Most security administrators will favor actually doing *both*, since there's no point leaving the WWW server exposed on all its ports if it's a production system and we want to secure it. So imagine we install a fairly strict access control policy that allows only port 80 traffic from the outside Internet to the WWW server, and return traffic.<sup>3</sup>

Suddenly, we've completely changed the value of our NIDS' data! Now, if we see any traffic on a port other than port 80, it's a red alert because we can infer that the perimeter router is not doing its job anymore – especially if the traffic is outgoing from the WWW server! But, to the NID, port 80 traffic is still interesting and there's plenty of it to look at. Typical IDS usage would proceed along this path, but we might further turn out noise if, for example, we didn't want to be notified about CodeRed anymore. But there are really two layers of tuning in effect here – a broad policy-based "tuning" performed at the boundary router, and some detailed tuning in the NIDS' HTTP signature set.

Now, let's look at a more complex problem. The firewall is used to mediate access between the internal network and the Internet, for employee use. In order to allow basic user services, let's assume we configure the boundary router to allow all traffic aimed toward the inbound network to pass – we're going to let the firewall take care of it, and the NIDS will watch for attacks as the traffic passes. So our NIDS is now running a policy of "all signatures turned on for traffic heading to/from internal network and Internet, all signatures turned on except CodeRed going to/from WWW server."<sup>4</sup> Unfortunately, even though we have the NIDS *outside* the firewall, the firewall still interferes with the NIDS' ability to detect attacks.

<sup>2</sup> There are numerous variations among these three approaches, as well – including in-kernel host-based application aware approaches, host-based network-layer approaches, etc. We're not going to dwell on the terminology, relative merits or properties of these approaches.

<sup>3</sup> Most sites never get to install such a simple and effective policy, unfortunately, but this paper is not about firewall policy design.

<sup>4</sup> If we had a NID that let us do alert rewriting, we might enhance detection by making any non-HTTP alerts related to the WWW server a maximum priority.

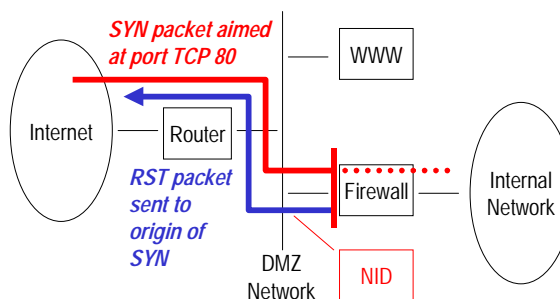
**Figure 3** illustrates how the firewall handles an incoming connection to a port that is denied by its policy. Someone on the Internet has tried to HTTP browse (TCP: 80) to a machine behind the firewall. The firewall receives the packet and sends back a reset packet to tell the remote system that the target is unavailable. The NIDS can track this transaction, but will never be able to learn if the connection represented an attack: the connection was terminated before the user revealed their intentions.

It gets worse: depending on the type of firewall, instead of sending a RST packet it might either send a network unreachable packet, or simply not send anything at all. In the case where it doesn't send anything at all, the NIDS sees a SYN packet with no RST or ACK. Depending on the NIDS' logic for detecting SYN flooding denial-of-service attacks, it might eventually conclude that the internal network was under a denial-of-service attack. It would be wrong. But is that a false positive? No. Really, it's more a case that the NIDS needs to be tuned for site-specific configuration.

Noise may also be generated if attacks get through to the firewall and it blocks them. For example, if someone launches a ping of death attack against a system behind the firewall, the firewall should correctly discard that traffic – but the NIDS should identify the packet-oriented attack. End users who care about getting an accurate reading of the noise level outside of their firewall need to have a solid understanding of which signatures are detectable by single packet transactions, and which signatures require a completely established TCP session to succeed.

In the author's experience, establishing an understanding of the properties of each attack is very time-consuming – many customers are getting telemetry from NIDS outside of firewalls: telemetry that is subtly wrong in ways that is product and site-specific. Similar issues exist with execution-blocking HIDS when used in cooperation with antivirus products or content filtering systems; it becomes a matter of which security subsystem acts first and how it acts. At present, end users should not expect IDS to be able to unravel these mysteries – a certain amount of tuning may be inevitable.

**Figure 3:**  
Firewall interferes with NIDS view by interrupting SYN



## Processing in the Console and Back-end Systems

The next trend in IDS is centralizing all the alert information from multiple IDS sensors. Sensors, in this case, may include HIDS, NIDS, system logs, and other data sources. Backend alert rewriting, normalization, compaction, and correlation allows the administrator to tune the IDS results in a single place, without requiring an administrative adjustment for each and every sensor. Alternative approaches combine tuning alerts at the backend with tuning signatures at the sensor level, to reduce the amount of traffic sent to the backend system.

Many vendors are able to export data from their IDS sensors or central repository into commercial SQL databases for further analysis. This is valuable for end users that wish to produce summary information regarding the types and severity of attacks seen within the enterprise; it's basically impossible to generate a broad-scale view from a single sensor. Accomplishing useful reporting will be an interesting challenge in the future, since it will involve combining low-priority noise with high priority information and reducing out instances where multiple sensors have identified the same event from different perspectives.

For the end user or product reviewer who is considering a test of enterprise-scale IDS tools, there is a lot of unexplored ground to cover. For example, what is the minimum useful configuration to test? Will one or two sensors be sufficient, or is counting sensors even useful? Perhaps the system should be measured based on its ability to cope with specified levels of alert traffic. The alert traffic should be sufficiently diverse and should contain a reasonable mix of noise to real alerts.

## Measuring IDS Effectiveness

Today, most IDS measures focus on two values: the number of attacks detected out of a known set, and at what level of utilization the IDS fails. Typical articles in the trade press or from testing labs read something like: "at 400Mb/s the IDS was able to detect 95% of the attacks launched against it." Two years ago, the main measure used to compare IDS was "which has more signatures" so the state of the art has improved considerably. Signature comparisons only make sense if all the IDS involved use simple pattern-matching signatures and don't do state tracking or protocol analysis. How many signatures can you count for a test that checks for an infinite number of possible buffer overrun attacks against a given web server?

Any answer between one and infinity is correct; signatures don't work as a measurement criterion. So the community has gravitated toward measuring ( **attacks detected / attacks sent** ) at various stress levels. The author believes that it is important to ensure that if such a measure is used, the tester should include a number of tests that are intended to deliberately trigger false positives and noise.

In other words, the tester may wish to:

- Replay recorded packets from a real CodeRed worm scan
- Replay recorded packets from a real CodeRed worm scan without the SYN packets needed to complete setting up the TCP connections
- Invoke a real CodeRed scan against a machine that is not vulnerable to it

The last item will be a crucial test, since IDS of the future appear to be likely to incorporate more detailed site configuration information. It may be necessary for the IDS (either the sensor or central management station) to be able to actively probe a potentially vulnerable machine, in order to either pre-load its configuration with correct information, or to perform a retroactive assessment. An IDS that performed some kind of actual system assessment would be a complete failure in today's generic testing labs, which focus on replaying attacks and scans against nonexistent machines. Most of the IDS testing methodologies in use today will be outdated in 2 or 3 years.

When performing the measurements above, the IDS should be scored on the number of alerts generated, the overall accuracy with which they match the attacks sent, the number of false positives, misdiagnoses, and amount of noise generated. Ideally, there should be a nearly 1:1 mapping, but since some attack tools represent a “bundle” of individual attacks, it may be the case that a single tool will generate 1 alert, or 50. The best case would actually be if the IDS were to correctly sort the real alerts from the noise as well as performing an accurate overall diagnosis. E.g.: “Seen: 1 TESO FTP Scan (consisting of 45 attack attempts), 2 alerts (vulnerable systems) 43 low-priority events (system not vulnerable)” The IDS test of the future will also need to consider a metric for tuneability of signatures and policy. Perhaps systems should be scored on a difference between pre-tuned results and site-tuned results, or at least include those values as a data point.

Most of the testing approaches that have been popularized in the commercial IDS industry are horribly flawed.<sup>5</sup> Some of the flaws are clearly deliberate and were intended to mislead buyers as to the performance properties of the products in question. For example, in one notable test, a testing lab generated synthetic background noise by producing 600Mb/s of TCP packets aimed at port 0. Most IDS will correctly discard this traffic since port 0 is not a valid port – the “benchmark” measured the IDS’ ability to throw traffic away.<sup>6</sup> Not surprisingly it was able to do this at a high rate of speed.

We’re beginning to see the beginning of a movement from such sham benchmarks toward more reasonable benchmarks, but many of the current benchmarks are still flawed because they rely on producing partial traffic intended to trigger known signatures in the IDS product being tested. One test was performed by taking a particular IDS’ signature database, pulling the patterns and port numbers from the signatures, then generating packets on those ports containing text that would match the patterns. Such a benchmark actually measures the IDS’ ability to generate false positives, if the traffic is not valid traffic on connected TCP streams! Some commercial IDS testing products have appeared on the market lately, which attempt to do a better job by more closely simulating real TCP streams.

As those products improve, they will be a valuable tool for testers. At present their integrity is questionable and the integrity of a lab using synthetic attack traffic generators is open to question. Another form of synthetic attack generator that should not be used is the vulnerability assessment tool. Vulnerability assessment tools usually perform enough of an interaction to identify the version of a piece of software running on a network, so they can compare the version with a vulnerability database. Their “footprint” running on a network is extremely different from the “footprint” of an actual attack tool. Some attack tools perform a version probe like the vulnerability scanners, but others simply launch a series of attacks against all known versions of a particular piece of software.<sup>7</sup> Most commercial IDS products have signatures that attempt to identify vulnerability scanner activity as “reconnaissance” traffic. This is a response from the vendors to the large number of early customers who used vulnerability scanners as the only testing tool for IDS.

Testing IDS under various load levels is still a worthwhile effort, since many IDS are quite load/performance sensitive. In order to get a realistic reading, it is absolutely essential that any load generators used should produce accurate simulations of real traffic. In one benchmark the author saw, a synthetic load generator was used that produced TCP sessions consisting of a SYN packet, a series of data/ACK packets, but no FIN packets. One IDS tested with this rig crashed due to resource starvation as it waited for the millions of sessions to close. Another passed the test with flying colors because it wasn’t performing state tracking at all. Neither product showed correct behavior, but the one that didn’t track state got better scores on the test.<sup>8</sup> If an IDS is able to identify the traffic as synthetic it is within reasonable design parameters to discard the traffic, invalidating the test.<sup>9</sup>

<sup>5</sup> See Ranum: experiences testing IDS ([www.nfr.com/publications/white-papers/Benchmarking-IDS-NFR.pdf](http://www.nfr.com/publications/white-papers/Benchmarking-IDS-NFR.pdf))

<sup>6</sup> Alternatively, the IDS should have generated 600Mb/s worth of alerts regarding all the illegal packets it was seeing.

<sup>7</sup> The Teso FTP mass-rooter is an example of the “sledgehammer” approach to exploiting vulnerabilities and it is instructive to compare its footprint to that of a vulnerability scanner.

<sup>8</sup> They both should have gotten an ‘F’ for state tracking.

<sup>9</sup> Why an IDS would discard traffic *silently* is a mystery – unless the vendor is doing it as a performance optimization. After all, an IDS is supposed to be identifying weird/hostile traffic on the network, and apparent TCP streams coming from nowhere, and going to nowhere is not normal by any stretch of the imagination.

IDS test designs should carefully separate the “background noise” aspect of the test from the “attack stream” test in their methodology. The author believes that the only valid way to test attack traffic is to have actual attack tools being run against actual vulnerable and invulnerable targets.

Because IDS is a performance-sensitive product, there has been an undue focus on performance rather than on correctness. Buying an IDS based solely on speed measures is about as smart as buying a car based on its top speed without ensuring it has brakes and can turn at speed. A better understanding of how the IDS performs in its accuracy of diagnosis (false positives, false negatives, accuracy of diagnosis, and noise) will serve our understanding of the merits of IDS’ much better than packets-per-second ratings.

## Summary

Many end users of IDS are unhappy with the results and complain of too many false positives. In reality, most IDS function fairly well, they just need a little tuning to filter out the majority of the noise. It’s possible that future IDS’ will do a better job of reducing the amount of noise they produce, but it will mostly be done in backend systems at an enterprise level. For the time being, IDS designers and signature-writers are trying to cope with contradicting goals.

The systems they produce need to be sensitive enough to perform well on poorly-designed benchmarks and demonstrations, but absorb enough site knowledge so they generate a minimum of noise. In other words, *we’re asking for noise and complaining when we get it*. The IDS of the future will “learn” more about what’s going on within the local network, and will improve its noise-reduction rapidly after installation. Today, it’s a process that the administrator needs to understand and take an active part in, if this exciting new technology is going to fulfill its promise.

## About the Author

Marcus J. Ranum is an independent computer and network security consultant who specializes in product analysis, implementation, and design. In 1997, Marcus founded Network Flight Recorder, Inc. (Later NFR Security) a leading manufacturer of IDS products. As CEO and later CTO of NFR, Marcus was responsible for product design and managing implementation and testing. He has been involved in a great number of IDS product benchmarks and tests, including every major IDS product on the market. Marcus is a frequent speaker at conferences, a popular lecturer/instructor worldwide, and has authored numerous white papers and books on computer security topics. His integrity and no-nonsense approach to problem solving has made him an indispensable resource for many FORTUNE 500 companies and several national governments. Marcus is holder of the TISC “Clue” award for service to the computer security profession and the ISSA Lifetime Achievement Award.

## About ICSA Labs

For over a decade, ICSA Labs, an independent division of TruSecure Corporation, has been the security industry’s central authority for research, intelligence and certification testing of products. ICSA Labs sets standards for information security products and certifies over 90% of the installed base of network firewalls, PC firewalls, anti-virus, cyrptography, and IPSec products in the world today.

ICSA Labs manages and sponsors security consortia that provide a forum for intelligence sharing among the leading security vendors of security products. In addition, ICSA Labs publishes surveys, security industry studies and buyers’ guides for computer security products.



1000 Bent Creek Boulevard • Suite 200  
Mechanicsburg, PA 17050  
Tel: 717.790.8100  
[www.icsalabs.com](http://www.icsalabs.com)