

*Intrusion Detection with Open Source Tools*



*Managing Security with*

# Snort *and* IDS Tools

O'REILLY®

*Kerry J. Cox & Christopher Gerg*

---

# Deploying Snort

Deploying an NIDS presents an administrator with some real challenges (apart from attempting to find a rational explanation for management on the return on investment for a security project). Installing and getting Snort up and running is just the beginning. You need to figure out what you want to watch, how you can watch it, and how to get meaningful information out of your effort.

Many of the obstacles to your NIDS deployment efforts are not technical at all. You might have to convince management that intrusion detection has value on par with the dollars and labor involved. Another, sometimes unforeseen issue is that an organization may have separate departments for network, server, and security administration—and communication between the groups may be poor.

Snort makes meeting these challenges a bit easier. Snort is free and will run on relatively low-cost hardware (it's unreal how inexpensive memory and disk have become!). The initial installation and configuration of Snort is fairly straightforward, and you can use my experiences and advice in this book (and the available support of the open source community surrounding Snort) to aid in the ongoing maintenance and administration of your IDS installation. While Snort won't magically get your different departments talking to one another, Snort sits as a passive listener on the network, needing little cooperation with the other departments to get installed and running. Once you call the server guys with notification that they may be suffering a security breach and it is confirmed, you will see communication improve quickly.

Spending time and care on the installation, initial configuration, and placement of Snort will reduce false positives, improve performance, and ensure that you are watching what is important. Let's look at the nontechnical challenges to deploying an NIDS (Snort, specifically) and then dive into the technical issues.

# Deploy NIDS with Your Eyes Open

While this book discusses strategies to make the installation, configuration, tuning, and administration of Snort as efficient and effective as possible, it is important to understand that running an NIDS is not as simple as plugging it in and watching. People often underestimate the labor involved with the ongoing maintenance of an NIDS (any NIDS, not just Snort). While you can minimize their occurrence, false positive alerts keep you busy confirming that they are, indeed, false. New signatures come out that detect the latest batch of Internet worms and they need to be reviewed, tuned, integrated, and distributed.

One of the challenges of using an open source application like Snort is that there are new versions fairly regularly. These new versions may have additional functionality that you want to use. The only problem is that sometimes this functionality causes older ways of doing things to change or be replaced (the portscan2 and conversation preprocessors being replaced by flow-portscan, for example). Test new versions and functions before upgrading. Sometimes new functions can introduce new bugs, too. Fortunately, open source testing of beta versions along with the cooperation and development done by Sourcefire (the company that sells the commercial version of Snort) eliminate most bugs before they make it into production code.

None of the previous discussion even touches on the challenges involved when you really find evidence that you are under attack or that you've been hacked. An effective security manual that includes a thorough incident response plan will pay dividends (of course, developing the plan takes time, too). The difficulty of getting those Balkanized departments to work together will certainly figure in to the fun, too.

All of these things can conspire to make you a very busy administrator. Is having a good awareness of what is going on in your network and on your servers worth the effort? In my experience, absolutely. Sticking your head in the sand and being ignorant of the harm being done to your organization is no way to run a network.

We talked in the introduction about the concept of defense-in-depth, where each device on your network plays a role in its own security and multiple strategies are employed to make catching (and stopping) attacks possible. An NIDS deployment will not be the big box of security that some people think they need to “have security” in their organization (almost every organization has a person with an MBM degree—Management By Magazine). There is no such thing as a single device that will secure your network.

NIDS is another layer of defense. It compliments your efforts in other areas, catching things that your other efforts miss. You still need to apply security patches to your software and systems. You still need to segregate Internet-facing systems to an isolated network (usually referred to as a DMZ). You still need to audit your system logs. An NIDS provides early warning that someone is probing you or that an attack is being attempted against your systems—you catch them when they are looking in

the window or jiggling the doorknob instead of catching them after they are inside the house (or not noticing them at all).

## Initial Configuration

Take your time with the initial installation and configuration of your Snort system. Be sure to try things out; there are many options and a great deal of functionality at your disposal. Almost any effective information technology project will start with an inventory—an NIDS deployment is no exception. A thorough understanding of the types of systems, their location, and the services they provide will allow you to make educated decisions about how to configure your sensors.

## Targeted IDS

If you have only Windows systems in your network, employing the rules that watch for attacks on Unix-based systems will only generate noise. If you are running Apache as your web server, eliminating the Microsoft IIS rules keeps Snort from alerting on attacks that would not affect your web servers. Look at the information on the Snort rule sets in the next chapter and the discussion of tuning Snort in Chapter 9. They will help you eliminate rules that watch for attack attempts that will only generate false-positive alerts.

Consider what you want out of your NIDS installation. The Internet is noisy with scanners, worms, scripts, and probes. Generating alerts on all this noise is of questionable value. Before the advent of all this activity, a probe was usually a genuine precursor to an attack. These days, it is nearly impossible to discern the early stages of an actual network probe from this noise. Tune your IDS to match the operating systems, software, and devices you are running in your network.

Some of the strategies presented in the next section can help you target your efforts more effectively.

## Sensor Placement

Since the Snort sensor can only alert on what it sees, the placement of the sensor is very important. In many networks, putting the sensor in the wrong spot can cause you to miss an entire network's traffic. Figure 6-1 illustrates this in a simplistic example. If you place the Snort sensor at point A, you will be able to see all traffic between the internal network and the Internet. You will not be able to see the traffic between your DMZ (containing a web server and mail server) and the Internet. In this case, an attack on your web server would go unnoticed.

If you put the sensor at point B, you will see all traffic between the systems in your DMZ and the Internet. In this case, you will not see the traffic between the internal

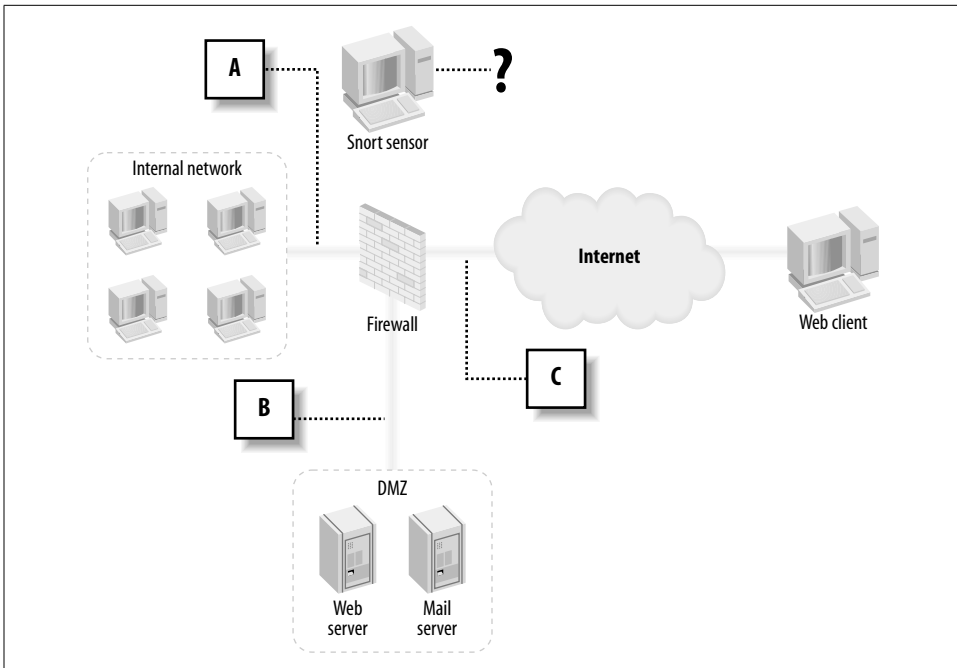


Figure 6-1. The importance of sensor placement

network and the Internet. Please note that this might be desirable. Perhaps you have a sensor dedicated to the DMZ with a tuned set of rules and preprocessors specifically for those servers. You might have another sensor located at point A that watches that traffic that is tuned appropriately.

Locating a sensor at point C will allow you to see all traffic traveling to and from both networks (DMZ and internal) and the Internet. Putting the sensor at point C still leaves a potential blind spot: traffic between the DMZ and the internal network will not be watched.

As you can see, connecting your sensor to the network is not a trivial decision. Let's examine some of the aspects of this decision.

## Systems and Networks to Watch

It is not realistic to expect to watch all traffic between all systems on your network effectively with NIDS. Prioritization of your systems and networks is necessary.

Systems that provide services to the Internet are a good first choice. These systems are more at risk than systems on your internal network. They also may be providing services to your customers or business partners that are very important (if not increasingly central) to the goals of your organizations. A rule of thumb is to

segregate any systems that provide services to the public Internet in a separate network that has limited access to your internal network. This arrangement makes watching the traffic much easier.

There are also a group of servers that provide services to people sitting at the desks: print servers, file servers, authentication and directory servers, mail servers, intranet servers, and databases. The incidence of false positives increases greatly when watching internal LAN traffic. The normal chatter between Windows systems can generate an overwhelming volume of such alerts and the sheer amount of effort involved in tracking these down outweighs the value of doing so. Target high-value systems—the database storing your ERP solution or your accounting systems—for NIDS sensor vigilance. You can make up for the lack of an NIDS watching traffic by the disciplined administration of your systems, including following best practices when building the systems, pervasively using antivirus software, and auditing system logs.

This isn't to say that we will ignore the workstations, laptops, and other citizens of the internal network. It is suggested that traffic between these systems and the Internet be watched by an NIDS. If you have a WAN connection to your business partners or branch offices, a sensor watching traffic across these links is advised.

The exact placement of sensors is made easier by looking for natural bottlenecks—connections between networks make very nice connection points. The point (or points) that your network connects to the Internet is an easy choice. As previously mentioned, WAN links are important bottlenecks to watch. Consider putting your internal servers on a separate network so that traffic between the networks containing your desktop users and your servers can be aggregated and watched.

One question remains: do you want to place your sensor on the Internet side of the firewall (outside) or on the internal side (inside)? Outside, the sensor sees all inbound attempts to probe the network. Inside, it provides even more targeting for your sensors, since only inbound traffic that makes it through open ports in the firewall will be monitored. Given the noise and the questionable usefulness of alerting on that noise, I prefer locating the sensors behind the firewall.

## Creating Connection Points

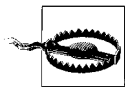
The high-speed switches that most networks use direct traffic to just the ports that connect to the systems engaging in the conversation—making eavesdropping on that conversation with an NIDS sensor impossible without further assistance. Some administrators plug a small hub into the path of the traffic they want to watch. While this tactic works (hubs send all traffic out all ports), small hubs are very often not as reliable as enterprise-class switches.

There are a variety of hardware network taps that can make a copy of traffic traveling on copper and fiber network cables. Some are commercial and others can be built with instructions on the Internet. The commercial solutions are usually well made,

but represent a potential single point of failure for the cable being tapped. While I know one end of a soldering iron from the other, I wouldn't want to trust my network's Internet connection to my skills.

A better solution is available from most enterprise-class switches on the market. The specific examples here are those used on Cisco switch hardware. It is possible for the switch itself to make a copy of the traffic from one or several ports and send it out a designated port that you can plug the NIDS sensor into. This is called a SPAN port (Switched Port Analyzer) by Cisco. You can mirror the traffic from a single port, or aggregate the traffic from multiple ports (even ports on remote switches!). Please note that we are talking about Cisco switches, not routers.

Because you are using your existing (likely redundant) hardware to perform this task, you are not introducing any potential failure points. Going back to Figure 6-1, if we create a SPAN port that aggregates the traffic from both points A and B, we can watch traffic between both the internal network and the Internet and the DMZ and the Internet with a single sensor interface.



Plan carefully. On Cisco switches, you are allowed only two SPAN ports per physical switch.

There is one thing to watch out for when spanning multiple ports into a single monitor port. If the total bandwidth you are aggregating is larger than the bandwidth of the port you're sending it to, the extra traffic will be lost and not monitored. For example, if you are sending the traffic from five 100 Mbps Fast Ethernet ports—each of which is carrying 25 Mbps of traffic—to a single 100 Mbps Fast Ethernet port, you will lose 25 Mbps of traffic, since you cannot send 125 Mbps of traffic through a 100 Mbps port. When possible, I try to use a Gigabit port as the SPAN port to get around this problem. In truth, this much traffic causes most systems running Snort to get little Xs on their eyes and fall over backwards. For such high traffic levels, consider some of the strategies presented in Chapter 13.

## Encrypted Traffic

It's also important to have a good understanding of how your systems perform their business. You might want to use Snort to monitor a very important e-commerce web server that processes credit card purchases from Internet based customers. This traffic is encrypted using SSL encryption—making the transaction much more secure. It is impossible for the Snort sensor to match the content of an encrypted packet to signatures in the rule files. The goal of encrypting the traffic is to make it nearly impossible to intercept and monitor. Figure 6-2 illustrates this situation.

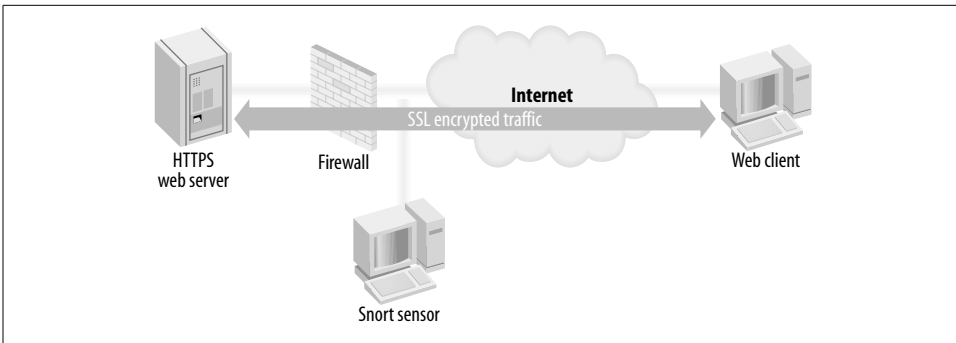


Figure 6-2. Snort can't watch encrypted traffic

One solution that allows your web traffic to remain encrypted (maintaining the security of the transaction) but also allows Snort to watch for signs of intrusion is an *SSL proxy*. SSL proxies go by many names, including Content Switch, SSL Accelerator, and SSL Proxy. This device sits between the client and the server and handles the task of encrypting the traffic. The traffic from the web server to the SSL proxy is not encrypted, but the traffic between the proxy and the web client *is* encrypted. Plugging the Snort sensor between the web server and the proxy allows the traffic to be monitored. Another real advantage to the implementation of an SSL proxy is the ability to offload CPU-intensive encryption tasks to a dedicated device, allowing your web servers to scale more effectively. SSL proxies very often can perform other tasks, such as load-balancing and authentication. Figure 6-3 illustrates the implementation of an SSL proxy between the web server and the client (and where the Snort sensor can be connected).

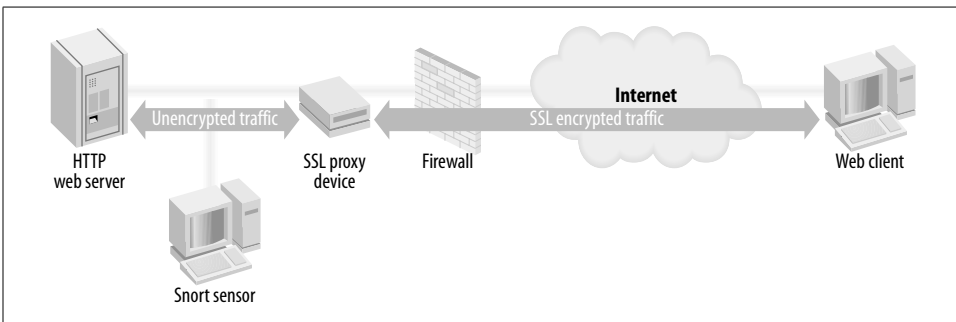


Figure 6-3. Snort is now able to watch the web transaction

## Securing the Sensor Itself

It should be obvious that protecting the integrity of the systems responsible for monitoring and maintaining the security of your network is very important. You need to protect the integrity not just of your NIDS systems but of your syslog servers,

authentication servers, monitoring, and management tools. One strategy that I employ is a *management network*. This network is behind its own firewall and access to the systems contained within the management network is closely controlled. The systems inside do not even participate in the same authentication domains as the systems on the inside of the network. The only openings in the firewall are those that are needed to get monitoring traffic to the systems that watch the environment.

Closely managing the Snort systems is important. The operating systems should be configured according to industry-accepted best practices and should be kept up-to-date with patches and updates. After all, a compromised IDS sensor would have access to all the traffic to and from your most sensitive systems—a dangerous situation, to say the least.

## Choose an Operating System

I squirm uncomfortably when people ask me what operating system they should use for a particular function. I carefully try to avoid discussions about religion, politics, and operating systems—all for the same reason. They're too controversial. However, I will briefly put on my consultant hat and walk you through the process of deciding which operating system to use for your Snort sensor.

I started my Information Technology career doing Microsoft phone tech support for the launch of Windows 95. My background in modems and networking (Commodore 64, 300 baud modem at the beginning of things) quickly got me into a mentor role. My MCSE certification was first in NT 3.51. I consider myself a Windows expert. That said, about five years ago I discovered Linux and BSD. I liked the fact that I could strip the system down to just the services I wanted. It made administration much easier (and the side effect was better general security). I hated the fact that my Windows servers needed to be rebooted all the time to keep them running. Linux had outstanding uptime.

Now, if I build a system, it's a Linux box (RedHat or Suse are nice) more often than not. My desk at work has three systems: a Windows XP laptop, a Suse Laptop, and a FreeBSD server. At home I have a Debian server (my mail server), a Suse linux server (test box), a FreeBSD server (Web Server), and a Windows gaming system. I spend about 60% of my time in a Unix-based environment of some sort. If I am going to build a server, it will be either Debian or FreeBSD, since I can build a minimum configuration and keep it up to date very easily. (I really like FreeBSD, but my company has standardized on Redhat as a Linux server OS). The recent changes in Suse are pretty exciting, too.

I share this information to give you a frame of reference for my approach to choosing an appropriate operating system. There are several things to consider; supportability, performance, stability, and security are at the top of the list:

### *Supportability*

Very often, the decision of which operating system to use is based on what you know how to run. Choose an operating system that you know how to configure and maintain effectively. If you know Windows well and little or nothing about Linux, use Windows! I'm speaking about a Snort system that will be relied on to secure a network. If you are just playing or testing, installing and configuring Linux or FreeBSD Snort sensor might be a nice way to learn a new OS. Most of the web resources dedicated to running Snort lean towards a Linux-based installation. It is arguably easier to "strip down" a Linux or BSD system—Windows installs a lot of things you just don't need.

### *Performance*

It is generally accepted (and I'm opening a real can of worms here) that the network stack on Linux and BSD is faster than the Windows network stack. In my experience, it seems that a Linux sensor can watch higher bandwidth levels than Windows using a given configuration. This makes sense when you consider that Snort (and the underlying infrastructure of libpcap) is written for a Unix-based operating system.

### *Stability*

It used to be an easy argument that Linux and BSD were much more stable than Windows. That really isn't as true anymore for a well configured and patched Windows system. I would still argue that a competently administered Unix-based operating system has better uptime compared to Windows (email me with your arguments). When considering stability, you may find yourself going back to the supportability issues.

### *Security*

I can lock down and secure a Windows system as well as I can lock down a Linux or FreeBSD system. The problem is, it has been much (much!) more work to keep a Windows system secure over time. The number of patches released for Windows services has been nearly overwhelming. There have certainly been patches for other operating systems and Unix-based services. However, there have been fewer in general, and since Unix-based operating systems tend to run fewer services, there is a smaller chance that you will be affected by a particular vulnerability. For security-related tasks, I tend to migrate to Linux or BSD over Windows.

## **Configure Interfaces**

Along with the creation of a controlled management network, there are more steps to be taken to protect the integrity of your security systems.. Snort sensors should be configured with at least a pair of interfaces. One of these interfaces will be on the management network; all alerting and management traffic will use this interface, keeping it away from prying eyes. Snort will use the other interface as a monitoring

point. This interface will not be configured with an IP address, so it will be invisible to hosts on that network. This is commonly referred to as a *stealth interface*. Keeping the listening interface invisible to the other systems on the network makes keeping the sensor secure much easier.

We discussed using SPAN ports in a switched network to allow the monitoring of switched ports. SPAN ports can actually help hide your Snort sensors, since they can be configured to only transmit traffic from monitored ports and not listen to interfaces plugged into them.

## Disable Unnecessary Services

If a service is not needed for the business function of a server, it should not be installed or enabled. The fewer services running on a system, the fewer potential issues need to be secured and kept up-to-date. This is an essential step to making a system secure.

## Apply Patches and Updates

These days, there are always updates and patches that need to be applied to the operating system and services—especially on a freshly installed system. This is true no matter what operating system you use. As time goes by, it is important that administrators keep abreast of newly discovered vulnerabilities in their operating system and services. If possible, establish a maintenance window for when updates will be made to your systems. This will allow you to establish a change routine—notify necessary people that the system will be unavailable, test the updates on a test system, and establish a back-out plan so that if the change blows up, you can go back to the initial system state.

## Utilize Robust Authentication

Where possible, use stronger authentication than just a simple username and password. The weaknesses inherent in passwords are well-documented. Passwords can be attacked through dictionary attacks or simple brute-force. If you do need to use passwords, utilize mechanisms (varies from operating system to operating system) that enforce passwords of a certain length and complexity. Force the users to pick different passwords by remembering the last few they've used. Force the passwords to be changed periodically (every 60 days or so). Most importantly, configure the authentication mechanism to lock out the account after a certain number of consecutive failed password guesses (I prefer locking the account after five failed guesses).

If you can, employ a stronger mechanism for authenticating users (it's not possible in most environments). Smart cards (and PKI), one-time password generators, or biometric mechanisms are excellent choices.



I do have some reservations about biometric authentication. While very convenient and certainly better than passwords, this method has one serious flaw. When someone loses their smart card or one-time password-generating device, the old one is marked as revoked in the authentication system and a new card or mechanism is issued. If someone intercepts or finds a way to decipher the digital representation of a biometric authentication, how can I revoke a thumb or a retina?

## Monitor System Logs

It is very important that the system is configured to generate logs and that those logs are reviewed regularly for signs of system, hardware, or configuration problems (including signs of intrusion). Auditing authentication, system function, and hardware operation is a good place to start. If possible, send the logs to a central syslog server (hopefully located on your controlled management network). This makes it much easier to review the logs and establish some correlation of events across multiple systems and networks.

## Using Snort More Effectively

It doesn't take long before only logging to the alert file becomes ineffective. The alerts scroll by too quickly and making sense of the data logged in a timely manner can be impossible. In Chapter 5, we looked at how to configure Snort to log to a database. The information sent to the database contains an incredible amount of information—including details about the packet that triggered the alert. Refer to Appendix A for details on the data contained in the database. Choose a database and configure Snort to send alerts to it.

Once the data is in the database, you need to choose some tools that can present the data in a way that makes managing the alerts and the sensors quick and easy. I prefer to use ACID (the Analysis Console for Intrusion Detection). You may find that another tool suits you better. Refer to Chapters 10, 11, and 12 for an examination of the tools that are available to help you manage your Snort-based NIDS deployment. Chapter 7 discusses strategies to keep your signatures up-to-date and effective.

## Sites of Interest

Guide to Hardening Linux

<http://www.antioffline.com/deviation/linux.html>

Guide to Hardening BSD

<http://www.antioffline.com/deviation/bsd.html>

Apache 1.3 Security Tips

[http://httpd.apache.org/docs/misc/security\\_tips.html](http://httpd.apache.org/docs/misc/security_tips.html)

Apache 2.0 Security Tips

[http://httpd.apache.org/docs-2.0/misc/security\\_tips.html](http://httpd.apache.org/docs-2.0/misc/security_tips.html)

MySQL Security Tips

<http://dev.mysql.com/doc/mysql/en/Security.html>

NSA Secure Configuration Guides

<http://www.nsa.gov/snac/index.cfm?MenuID=scg10.3.1>

Microsoft Central Security Portal

<http://www.microsoft.com/technet/security>