



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at <http://www.giac.org/registration/gsec>

Using Snort as an IDS and Network Monitor in Linux

James Kipp

June 13, 2001

Introduction

This paper will describe the benefits of using an Intrusion Detection System (IDS) in general, and Snort specifically. This paper will also describe techniques to use Snort effectively and some of the interesting features. It will also look at the importance of interpreting the data that Snort outputs with some examples of packets captured by Snort. In order to get the most out of Snort or any other Intrusion Detection System it is imperative to be able to intelligently interpret the data (packet captures) the program logs. This paper will use specific examples on some of the more useful features of Snort. It will not be a how-to on installing, configuring, and running Snort as there are many documents on the Internet that cover that in detail and other GSEC papers that cover it. This paper will be slanted toward the home user or small business networks. Some links on installing, configuring, and deploying snort will be provided in the notes section of this paper.

Overview

An Intrusion Detection System is aptly named. Robert Graham provides a good general definition of an IDS: "monitors packets on the network wire and attempts to discover if a hacker/cracker is attempting to break into a system (or cause a denial of service attack). A typical example is a system that watches for large number of TCP connection requests (SYN) to many different ports on a target machine, thus discovering if someone is attempting a TCP port scan. A NIDS may run either on the target machine who watches its own traffic (usually integrated with the stack and services themselves), or on an independent machine promiscuously watching all network traffic (hub, router, probe). Note that a "network" IDS monitors many machines, whereas the others monitor only a single machine (the one they are installed on)." [1]

Intrusion Detection systems can be passive or active. A passive system simply logs and alerts the administrator of unusual events or traffic entering the system. It does not act on the event or change any system configuration. With a default installation, Snort is a passive IDS. Snort monitors or "sniffs" network packets and logs and can alarm the administrator of a packet matching a specific rule or containing specific information. However, there is a compile time option and a 3rd party tool to force snort cause system changes, such as adding rules to ipchains (Linux packet filtering utility), in effect becoming an active IDS. This paper will explore the compile time option in detail as well as 3rd party tools. An active IDS has options of acting on unusual or flagged events (events deemed as an intrusion). It can be configured to block the source of the anomaly or make a system configuration change. Examples of active IDS are Portsentry and Dragon IDS. Intrusion Detection Systems are often used to supplement firewalls in protecting a system or a network. IDS is not the same as a firewall. It does not block or redirect network packets based on information found in the packet headers like a firewall does.

Snort is a light-weight intrusion detection system. It is not resource intensive and the source code is small. Snort is best used on small to medium sized networks, single hosts, or on segments of a large network. It is not recommended to be used to monitor an entire large scale network. Snort can be configured to be a packet monitor (sniffer) to capture and log all network packets coming across the specified network interface.

Rules

Most IDS packages have the ability to customize rules necessary for a particular network. Rules define what an administrator is looking for when packets cross his wire. This is where Snort excels. Snort has a very comprehensive, flexible, and not too difficult rule writing feature. There are many great pre-written rule sets available from the Snort web site and other web sites. An administrator can write his/her own rule set or combine rules with the pre-written sets. The following is a general description of the format of rules from Martin Roesch's paper "Writing Snort Rules". "Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information. The rule option section contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken." [2]

Let's examine a couple of rules :

```
alert tcp ![192.168.1.0/24,10.1.1.0/24] any -> [192.168.1.0/24,10.1.1.0/24] 111 (content: "|00 01 86 a5|"; msg:
"external mountd access");
```

The first word (alert) is an action keyword, which tells the rule how to react when it finds a matching packet. In this instance an alert will be generated using the user defined alert method and the packet will also be logged. The next word (tcp) is the protocol to look for. The next section containing the CIDR address block is the source of the packet. In this case the ! negates the defined source or states NOT from this source address. The "any" keyword means any IP address in the specified CIDR block. After the -> is the target IP address or CIDR block followed by a port number. The port number has many options as well such as "any" or a range of ports. The next section, in parenthesis, is the rules options. It starts off with an option keyword then the argument. More than one option may be used in a rule. The above rule has two options. The first is "content" which simply looks for specific content in the packet and forces a response to that content being matched. The content can be plain text or binary data which can be quite complex. In this case the content

match causes a message signified by the key word "msg" which specifies a message to print to the logs in addition to the packet capture information. Let's examine one more rule:

```
alert tcp any any -> 192.168.1.0/24 80 (content: "cgi-bin/phf"; flags: PA; msg: "CGI-PHF probe");
```

This rule tells Snort to alert and log tcp packets from any IP address and any port going to the 192.168.1.0/24 CIDR block of addresses on port 80 containing text "cgi-bin/phf" in the packet payload and flags Push or Ack in the TCP header. Finally, if it matches the rule write the message "CGI-PHF probe" to the logs with the packet capture dump.

Some Specific Features of Snort

Snort can seem somewhat overwhelming to the novice user because of the myriad of configurations and options available. But it is this flexibility which makes it such a useful tool and the time spent learning to use it is time well spent. It does require a working knowledge of TCP/IP and packet structure, so it may be wise to have a good TCP/IP book on hand such as Richard Stevens's "TCP/IP Illustrated, Volume 1: The Protocols". Snort can detect a wide range of known vulnerabilities and is constantly being updated to keep up with newer vulnerabilities and attack signatures. Snort is particularly adept at analyzing and parsing packet header and payloads and is great at filtering options placed in TCP packet headers. Let's look at a couple of features that snort can handle.

- **Fragmentation:** When writing a snort configuration / rules file, it is common to define variables and preprocessors. Preprocessors are directives that examine the packets before the actual rules are applied. It can be used to filter out packets that you don't want being processed by Snort or to modify parts of the packets before being analyzed by Snort rules. One useful preprocessor is the defrag or minifrag preprocessor. This allows you to specify and flag packets that are fragmented in general or fragmented under a certain size. Many attacks or scans will try to send small packets to get through a firewall. A preprocessor is written like this: "preprocessor minfrag: 128" [3]. This will flag fragments that are 128 Bytes or less in size.
- **Portscans:** One can also define a preprocessors to watch for portscans. You can tell Snort to ignore portscans from certain hosts or watch for portscans from certain hosts. It will log the type of scan and what hosts and ports were the target of the scan. Needless to say, this is a very useful feature. With 3rd party tools, an administrator can even force Snort to take action against the source address of the portscan. Here is an example of a portscan preprocessor: "preprocessor portscan: 192.168.1.1/24 3 5 /var/log/snort_portscan.log " [4]. This directive simply says to look for scans coming from 192.168.1.1/24, with a minimum of 3 ports being connected to and at 5 second intervals, and log it to /var/log/snort_portscan.log .

3rd Party Tools Used with Snort

The popularity of Snort has spawned an abundance of fine 3rd party tools to be developed. These tools range from processing and formatting the snort log files, to enabling Snort to work with other applications such as ipchains or tcp wrappers, and to log Snort directly to a database. These tools add to the efficiency and reliability of Snort. This paper will discuss a few of these tools. For a long list of tools check out <http://www.snort.org/snort-files.htm>. One useful tool is **Guardian**. Guardian will use ipchains to build a rule dynamically based on what Snort finds. Ipchains is the utility used to build Linux based packet filtering firewalls and NAT. For instance, you may have Guardian write an ipchains rule to block packets from any hosts that tries to send packets containing certain information. This essentially turns Snort into a active IDS. **Snort2html** is a Perl script that converts Snort alerts and logs to HTML web pages, which an administrator can view from a website (if there is a Web server setup) or just have the HTML document emailed. **ACID** is a popular reporting tool for Snort [5]. It has a user friendly interface and has a multitude of features to help you get the most out of Snort. With ACID, you can search for alerts based on criteria you specify. You can configure to whom and where alerts should be sent. You can even use it to filter out false positives. False positives are alerts or logs that and IDS may flag that is probably innocuous. A false alarm if you will. These are three popular snort tools, but there are many more fine tools to choose from.

Examining Snort Output

Now the fun begins. You have your Snort IDS running and catching all of the bad guys packets coming at you. You have it setup to email you with alerts and you have a nice 3rd party tool to display the log files in a friendly format. But to really get the most out of snort, you need to know how to interpret the output. This is where a good book on TCP/IP comes in handy, like the Richard Stevens books and "Internetworking with TCP/IP" by Douglas Comer. Let's look at a few examples of some Snort output. These are simple examples, but you may receive some very complex data that you will have to research to fully understand it. The first example is a packet dump that was generated by Snort running as a sniffer, basically just grabbing packets off of the wire:

```
06/11-15:33:48.918647 192.168.1.17:3128 -> 192.168.1.1:53 UDP TTL:64 TOS:0x0 ID:22715
Len: 47
```

Note: Spaces have been added between the fields for clarity. First we have the time/date stamp. The second field is the source IP address of the packet and the port number followed by the target IP address and port. Next is the protocol. In this case it is a UDP packet. Since it is UDP we can eliminate all of the TCP services and by checking common ports we know that port 53 is the DNS port, so this is some sort of DNS request. Next is the current Time To Live of the packet. Next is the TOS or type of service field. This field is basically options set in the packet to modify how it might be routed to it's destination. The value here "0x0" means normal service, or no options set. Other possible values are:

```
Minimize delay: 0x10
Maximize throughput: 0x08
```

The next field is the packet ID and finally the Packet Length in bytes.

```
052499-22:27:58.403313 192.168.1.4:1034 -> 192.168.1.3:143  
TCP TTL:64 TOS:0x0 DF  
***PA* Seq: 0x5295B44E Ack: 0x1B4F8970 Win: 0x7D78  
90 90 90 90 90 90 90 90 90 90 90 90 90 EB 3B .....;  
5E 89 76 08 31 ED 81 C9 31 C0 88 6E 07 89 6E 0C ^v.1.1.1..n..n.  
B0 0B 89 F3 8D 6E 08 89 E9 8D 6E 0C 89 EA CD 80 .....n.....  
31 DB 89 D8 40 CD 80 90 90 90 90 90 90 90 90 90 1...@.....  
90 90 90 90 90 90 90 90 90 90 90 90 E8 C0 FF FF FF .....  
2F 62 69 6E 2F 73 68 90 90 90 90 90 90 90 90 /bin/sh [6]
```

Using "Flexible Response" to make Snort an "Active" IDS

The "Flexible Response" option can be used to disconnect or drop the route of the source address of the offender. By adding rules to the current rules file, there are several nifty ways to disconnect from the offender. The simplest is sending a reset packet to just kill the connection. ICMP unreachable packets can also be sent to the offender along with the reset packet to make it look like our host is not online. We can even send packets to kill all current connections that Snort has flagged.

Even though this option is still in the testing stages, it is easy to see its potential value. Firstly, we don't have to rely on 3rd party tools to cause Snort to react to alerts it finds. Secondly, the options are more flexible than with the Guardian tool. However there is some drawbacks to using this option and the Guardian tool. An attacker could be spoofing the IP address to make it look as if the packets are coming from internal hosts or trusted hosts. This could disable the Snort box's connection to important hosts like its DNS server for instance. If you use this option, make sure you tell Snort in the configuration file or in your ipchains rule set to ignore or accept packets coming from these hosts. Also be sure to have a good way of alerting yourself of snort "alerts" by email or setting up a paging system.

Snort is an excellent choice for a small to midsize network or the home user, especially with a "always on" connection such as cable and DSL. Snort performs better than many of the IDS packages that cost money, lots of money. It does have a learning curve but most good tools do in my opinion. Administrators that can master Snort and update it frequently can substantially reduce or negate harmful attacks against the system or network. The developers of Snort also do an excellent job of maintaining and upgrading the application and there is plenty of documentation out there to help you. There is also a few informative mail list dedicated to using and configuring Snort.

- [1]) "FAQ: Network Intrusion Detection Systems" by Robert Graham March 21,2000
<http://www.robertgraham.com/pubs/network-intrusion-detection.html>
- [2] "Writing Snort Rules" by Martin Roesch http://www.snort.org/docs/writing_rules/
- [3] "Network Intrusion Detection Using Snort" by Dave Wreski & Christopher Pallack
http://www.linuxsecurity.com/feature_stories/feature_story-49.html
- [4] "Snort Installation and Basic Usage" by Dale Coddington
<http://www.securityfocus.com/frames/?focus=linux&content=/focus/linux/articles/linux-snort.html>
- [5] "ACID Project Homepage" Roman Danyliw <http://acidlab.sourceforge.net/>
- [6] "Snort - Lightweight Intrusion Detection for Networks" Martin Roesch
<http://www.snort.org/lisapaper.txt>

Helpful Links and Books

- 1) Richard W. Stevens. "TCP/IP Illustrated, Volume 1". Addison Wesley Longman, Inc, 1994.
- 2) "Snort - Lightweight Intrusion Detection for Networks" by Martin Roesch
<http://www.snort.org/lisapaper.txt>
- 3) "Installing Snort 1.6.3 on SuSE 6.x-7.x" by Dr.SuSe
<http://www.linuxnewbie.org/nhf/intel/security/snort.html>
- 4) "Snort Documentation" by Andrew R. Baker
<http://www.dpo.uab.edu/~andrewb/snort/snortdoc/snort.html>
- 5) Douglas C. Comer. "Internetworking with TCP/IP" Prentice Hall, 2000.

© SANS Institute 2000 - 2005, Author retains full rights