

Snort Installation and Basic Usage Part Two

Hal Flynn 2000-07-31

I. Introduction

Part I of this article focused on the installation and basic usage of the snort intrusion detection system (IDS) on the Linux platform, including running snort as a command line sniffer and loading snort with a pre-defined rule set. This article will take a look at some further methods and programs that can be used in conjunction with snort to more reliably detect and fend off intrusions. We will also examine how rules are written to suit special case scenarios.

II. Update

Since Part I of this article was released snort has been updated. The most current version as of this writing is 1.6.3 and as always is available from www.snort.org. It is recommended that those using earlier versions upgrade to this latest version.

Among other things, v1.6.3 fixed a compilation problem on all non-BSD systems.

In Part I it was said that at this time Linux is not capable of performing packet loss statistics, while other Unix platforms. It turns out this is not entirely true. Phil Wood has since informed me that it is indeed possible to do so albeit with some effort. Mr. Wood stated:

"Alexey Kuznetsov made it possible to extract a dropped packet count on Linux using a system kernel configured with option CONFIG_PACKET_MMAP in conjunction with a modified linux-pcap.c. This permits the use of a ring buffer using shared memory which allows the libpcap based program to peruse packets on a ring while the kernel puts them on the ring if there is space. A simple flag cleared by the application and set by the kernel for each packet slot allows for management of the ring. If a program is not able to keep up, then the kernel will start decrementing a drop count. A BPF filter set by the application is used by the kernel to decide whether the packet should be put on the ring."

III. Rules

The first thing we will do is take a closer look at rules and how they are interpreted by snort. Following is a rule from the sample rule set we used in Part I:

```
alert icmp !$HOME_NET any -> $HOME_NET any (msg:"IDS152 - PING BSD";
content: "|08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17|"; itype: 8;
```

```
depth: 32;)
```

The first part of the rule set contains the rule's action (alert), protocol (ICMP), and also source and destination IP address and port information. This is known as the "rule header". The rest of the rule set, known as the "rule option", contains alert messages and information on what should be inspected in a packet to see if the rule matches. To get a better idea of what information is being looked at in the packet we will take a look at a trace of a ping:

```
07/23-09:46:41.866911 192.168.1.10 -> 192.168.1.1 ICMP TTL:50 TOS:0x0
ID:2403
ID:8474 Seq:256 ECHO
36 12 7B 39 1B C6 0B 00 08 09 0A 0B 0C 0D 0E 0F  6.{9.....
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  .....
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  !"#%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567
```

Notice how the content section for the rule above is actually contained in the trace sample above, starting at the ninth byte, and continuing through the 24th byte.

Using what we have learned so far, lets examine how snort uses the above rule. Snort examines packets going across a specified interface. Our rule is set to look for any ICMP traffic not originating on our monitoring machine "!HOME_NET" that is destined for our monitoring machine "-> HOME_NET". The "depth" in the above rule is set to 32. This means snort will search 32 bytes into each packet looking for the specified "content". If matching content is discovered, snort generates an "alert" logs the packet. The message contained in the rule will be logged to the appropriate log file "IDS152 - PING BSD". The "itype" is the ICMP type. In this case, it is type 8, which is an echo request.

Now lets take a look at another rule and examine how it works:

```
alert tcp !$HOME_NET any -> $HOME_NET 31337 (msg:"BACKDOOR
ATTEMPT-Backorifice";flags:S;)
```

This rule is also using the "alert" option, but this time is looking for "tcp" based traffic originating from outside the home network "!\$HOME_NET" and destined to the monitoring machine "-> \$HOME_NET" on a specified port "31337". If this pattern is detected, the message "BACKDOOR ATTEMPT - Back Orifice" will be logged in the system log file. The flags option in this rule is looking for a SYN. In a nutshell, this rule is looking for any traffic originating from outside our network and attempting to connect to port 31337.

Following is one more example rule:

```
alert tcp !$HOME_NET any -> $HOME_NET 21
(msg:"SCAN-SATAN-FTPcheck";flags:PA; content:"pass -satan");
```

This rule generate an "alert" if all the following criteria are met:

1. Traffic is generated from any IP address not within \$HOME_NET
2. Client connects to TCP port 21 (ftp)
3. TCP flags are set at "P"ush and "A"ck
4. Data sent from client contains "pass -satan"

If all these criteria are met the msg "SCAN-SATAN-FTPcheck" will be logged to the log file. This is obviously a scan from the program Satan trying to connect with the password "satan" on ftp.

The other types of alerts that are available are "log," which logs the packet but does not generate an alert, and "pass," which will ignore the packet. Snort currently supports three IP protocols: TCP, UDP, and ICMP.

As you can see from these examples, snort's rules are rather flexible. An in-depth paper is available on writing snort rules at www.snort.org.

IV. Responding to Scans

At compile time snort has an available option "--enable-flexresp" that enables "Flexible Response" code. Compiling with this option does require libnet (<http://www.packetfactory.net>). This code can be used to cancel connections on IP-level when a rules criteria are met. As an example, if the criteria of a particular rule are met, snort can send the sending socket a reset and kill the connection. For more information regarding this feature see the README.FLEXRSP that is included with the snort distribution. This code is currently alpha. Great caution should be exercised when using This code, as the potential for abuse does exist.

Anthony Stevens has written a program called Guardian (<http://packetstorm.securify.com/sniffers/snort/Guardian.tar>) that works with ipchains to deny further packets from an attacking host. Again, a program like this must be used very carefully, as an attacker could spoof packets as being from another host, and cause Guardian to insert rules denying access to the spoofed host.

V. Reporting

When run with the -s option, snort logs message to syslog. This can be especially useful when your running a program like logcheck (<http://www.psionic.com>). Logcheck can be easily modified to be run and parse all snort syslog entries into their own field in a logcheck report. Open your logcheck.sh file with a text editor and add the following to the section right after where the MAIL variable is set:

```
# Looks for all cases of snort entries in the logfile and
# puts them into a report

SNORT_FILE=/usr/local/etc/snort.entries
```

Now go down to the section of logcheck.sh where searches are performed and add the following:

```
# Check for snort scans
if [ -f "$SNORT_FILE" ]; then
    if $GREP -i -f $SNORT_FILE $TMPDIR/check.$$ >
    $TMPDIR/checkoutput.$$; then
        echo >> $TMPDIR/checkreport.$$
        echo "Snort Report" >> $TMPDIR/checkreport.$$
        echo "-----" >> $TMPDIR/checkreport.$$
        cat $TMPDIR/checkoutput.$$ >> $TMPDIR/checkreport.$$
        FOUND=1
        ATTACK=1
    fi
fi
```

Lastly, create the /usr/local/etc/snort.entries file. This should contain only one entry:

```
#Grab all instances of snort
snort.*
```

Now the next time you run logcheck snort will have it's own section in the report:

```
Snort Report
-----
Jul 23 22:34:15 dissent snort[8094]: BACKDOOR ATTEMPT-Back Orifice 2000:
192.168.10.10:22652 -> 192.168.1.1:8787

Security Violations
-----

Jul 23 22:34:15 dissent kernel: Packet log: inp DENY eth1 PROTO=6
192.168.10.10:22652 192.168.1.1:8787 L=44 S=0x10 I=63752 F=0x0000 T=56

General Information
-----
Jul 23 22:32:36 dissent tcplog: ssh connection attempt from foo.bar
```

There are other solutions available. One is snort_stat.pl (<http://xanadu.rem.cmu.edu/snort/>), a Perl script that parses the log file and generates a report that can be mailed to an administrator. snort_stat can even output the report in HTML format and be posted to a website. (This type of information should be kept in a protected directory on the web server.)

Following is a sample report generated using snort_stat.pl:

```
--Start sample snort_stat.pl report --

The log begins from: Jul 23 22:57:33
The log ends      at: Jul 23 22:58:
[4]->[4]
Total events: 5
Signatures recorded: 1
Source IP recorded: 1
Destination IP recorded: 1

The number of attacks from same host to same
destination using same method
=====

# of
attacks from          to          method
=====
5      192.168.1.10  192.168.1.1  BACKDOOR ATTEMPT-Back Orifice 2000

Percentage and number of attacks from a host to a
destination
=====

# of
%  attacks from          to
=====
100.00  5      192.168.1.10  192.168.1.1

Percentage and number of attacks from one host to any
with same method
=====

# of
```

```

%      attacks  from          method
=====
100.00    5      192.168.1.10    BACKDOOR ATTEMPT-Back Orifice 2000

```

The percentage and number of attacks to one certain host

```
=====
```

```

# of
%      attacks  to          method
=====
100.00    5      192.168.1.1    BACKDOOR ATTEMPT-Back Orifice 2000

```

The distribution of attack methods

```
=====
```

```

# of
%      attacks  method
=====
100.00    5      BACKDOOR ATTEMPT-Back Orifice 2000

```

```
-- End sample snort_stat.pl report --
```

As you can see this script can generate some useful statistics.

Another program available to generate HTML pages of snort entries in a log file is snort2html (<http://escert.upc.es/tools/snort/contrib/snort2html>). snort2html is also Perl script that will parse the log file and generate a report with all these entries.

Obviously, logs are only effective if they are generated and reviewed on a regular basis. Setting up a cron job to run these tools on as frequent as possible is strongly recommended.

VI. Conclusion

As you can see, snort can be extremely useful to detect intrusions on a network. When running a program like snort there are some things to be aware of. Running snort with the -l (log) option

generates a tremendous amount of logging information on a busy system. These log files should be examined on a regular basis. Also, rules can be subject to false alarms. Spend some time generating some attacks against the network you plan to run snort on. Make sure your rule set fits your needs. With a little effort rules for new attacks can be easily written. Be sure to keep your rules as current as possible, as a signature based IDS cannot spot attacks it does not know about.

Relevant Links

[Snort Homepage](#)

Martin Roesch

[PacketFactory.net](#)

Packetfactory.net

[Guardian](#)

Anthony Stevens

[Psionic](#)

psionic.com

[Snort Log Parser](#)

[snort2html](#)

snort2html

[Privacy Statement](#)

Copyright 2006, SecurityFocus