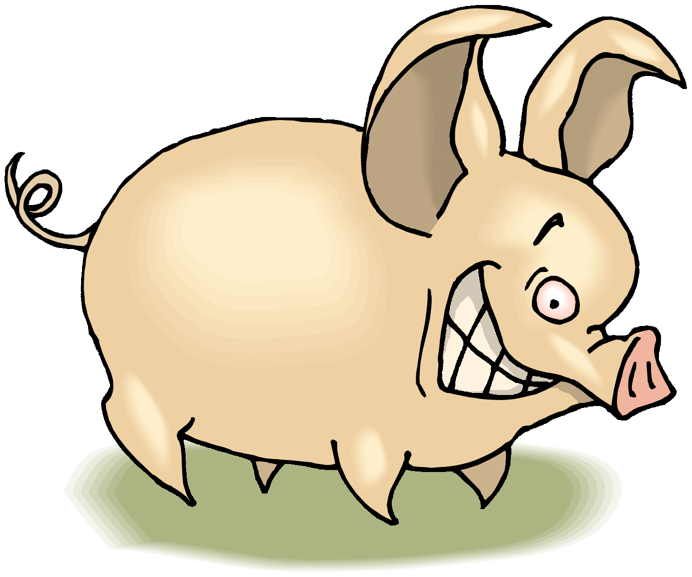

Introduction to Snort

A lightweight Intrusion Detection System by **Marty Roesch**

Document based extensively on original snort documentation from **/usr/local/share/snort**

Using Snort

What is snort?

Snort is a lightweight intrusion detection system that can log packets coming across your network. This program can be used on smaller networks but on larger ones, with Gigabit Ethernet, snort can become unreliable. Snort doesn't require that you recompile your kernel or add any software or hardware to your existing distribution but it does require that you have root privileges.

Why?

Many times it's far too easy for hackers to scan your network for vulnerable services that could be running or ports that are available. With this being a fact there isn't an excuse to ignore security when putting intrusion detection in is so easy to do. Having snort watch your internal network is important because many of the security problems actually come from inside your network and in that case you have more of a chance to correct something before it goes too far. Best of all this tool is free and available on most platforms.

When do I use snort?

Snort can be used anytime you want to have basic security measures in place that allow you to log and analyze the traffic on your network. Along with a firewall this should be another fundamental part of your network security.

How do I use snort?

Snort has a few uses: packet sniffer like tcpdump(1), packet logger for network troubleshooting, or an Intrusion Detection System. Snort can be placed on machines throughout your network and it works in promiscuous mode to watch all traffic on the wire. Snort can also be used to sift through already-made tcpdump files.

What platforms does snort run on?

Snort can run on Linux, BSD, MacOS X and Windows. Snort can run on all kinds of variants of these OSs and you can visit www.snort.org to find more information on it.

x86	Sparc	M68k/PPC	Alpha	Other	
X	X	X	X	X	Linux
X	X	X			OpenBSD
X			X		FreeBSD
X		X			NetBSD
X	X				Solaris
	X				SunOS 4.1.X
				X	HP-UX
				X	AIX
				X	IRIX
			X		Tru64
		X			MacOS X Server

Table from: http://www.snort.org/what_is_snort.htm

Snort Tutorial

The Output

The simplest way to start snort to see what is does is to use this command:

```
snort -v -i eth0
```

This command tells snort to be verbose and display the results to the console using the eth0 interface (Ethernet).

```
11/05-18:29:42.554864 192.168.1.2 -> 192.168.1.1
ICMP TTL:64 TOS:0x0 ID:275
ID:63745   Seq:0  ECHO

11/05-18:29:42.554962 192.168.1.1 -> 192.168.1.2
ICMP TTL:255 TOS:0x0 ID:2323
ID:63745   Seq:0  ECHO REPLY
```

The output contains a lot of information you may find useful. Here we can tell that we have an ICMP packet coming from 192.168.1.2 going to 192.168.1.1 and that the ICMP packet is an ECHO packet. This is the result of what you might find from a ping to this address. Then there's the ECHO REPLY from our machine sent to 192.168.1.2. The packets also contain the date stamp so you can see when they happened. When we stop snort using CONTROL-C we see this output:

```
===============================================================================
Snort received 6 packets and dropped 0(0.000%) packets

Breakdown by protocol:
    TCP: 0          (0.000%)
    UDP: 0          (0.000%)
   ICMP: 2          (100.000%)
    ARP: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 0          (0.000%)
===============================================================================
```

This can be really handy because it tells you your protocol statistics. Of course we have all of ours as ICMP.

Application Layer Data

Our next "trick" will be to display the application layer data as well. The application layer data is what is actually transmitted across the network. This can be used to sniff for passwords on a network as well because it displays the text that is passed between machines. You would do this by adding "-d" to the command:

```
snort -d -v -i eth0
```

Ethernet Information

If you want the Ethernet information to be displayed too this can be done using "-e":

```
snort -d -v -e -i eth0
```

Of course you can smash these options together to make it a little easier:

```
snort -dev -i eth0
```

ARP

Most of what we initially see are IP packets but we can add “-a” to the command to see ARP packets too:

```
snort –deva –i eth0
```

Logging

This is really interesting but not much more useful past real-time monitoring. We want to log these transactions to a file and that is done through the "-l" option accompanied by the directory you want to log to:

```
snort -dev -i eth0 -l $HOME/log
```

Make sure that you have or create the directory you wish to log to otherwise snort will complain! When packets are logged snort uses the address of the remote computer as it's logging directory.

Identifying Your Network

Sometimes you want to log packets relative to your network. To log packets into directories they are associated with use the "-h" option with the network address and mask of your home network.

```
snort -dev -i eth0 -l $HOME/log -h 192.168.1.0/24
```

Using Rules

Now for one of the most powerful parts of snort. SNORT RULES!. Indeed, snort would be limited if it didn't have a facility for rules. We'll cover rules syntax in a little bit but for now let's see how see specify what rules file to use. Snort comes with all sorts of built-in rules files in /usr/local/share/snort/. The rules files are simply ASCII files with lines of rules snort uses to determine what to log/alert. You'll notice that they can be quite simple or complicated but also cover some of the more sophisticated attacks and probes and they're pre-made for you. Snort rules can be specified with the simple "-c" option:

```
snort -dev -i eth0 -l $HOME/log -c rule-file
```

All together a snort command on the command line might look like this:

```
snort -dev -i eth0 -l $HOME/log -h 192.168.1.0/24 -c rule-file
```

This is quite a long command which is why this could be a good candidate for the old /etc/rc.d/ scripts files.

TCPDUMP Files

Another way to use snort is to sift through already created tcpdump files. Using the "-r" option you can start snort the way you normally would to check a tcpdump file for all the same kinds of things you use snort for on the network.

```
snort -d -h 192.168.1.0/24 -l $HOME/log -c rule-file -r tcpdump.file
```

Daemon Mode

Finally we want to find out how to put this thing into daemon mode so that it doesn't hog our terminal with the "-D" option:

```
snort -d -h 192.168.1.0/24 -l $HOME/log -c rule-file -D
```

Making Our Lives a Little Easier - Tips

Okay, now that we've covered most of the important snort stuff let's look at better ways to use it. For example, the "-v" could be left out because if snort is logging then you have that information anyway and the console would just be cluttered and for speed reasons it's really not required. The Ethernet "-e" data, too, is probably not really needed for most applications.

Introduction to Snort Rules

Now we'll take a look at some basic rules to see how they work. The basic format of a rule is as follows:

function protocol source_ip/mask source_port -> destination_ip/mask destination_port options

A rules file will contain many lines of rules and it can look a little complicated at first. Here’s an example.

```
# look for stealth port scans/sweeps
alert tcp any any -> $HOME_NET any (msg:"SYN FIN Scan"; flags: SF;)
alert tcp any any -> $HOME_NET any (msg:"FIN Scan"; flags: F;)
alert tcp any any -> $HOME_NET any (msg:"NULL Scan"; flags: 0;)
alert tcp any any -> $HOME_NET any (msg:"XMAS Scan"; flags: FPU;)
alert tcp any any -> $HOME_NET any (msg:"Full XMAS Scan"; flags: SRAFPU;)
alert tcp any any -> $HOME_NET any (msg:"URG Scan"; flags: U;)
alert tcp any any -> $HOME_NET any (msg:"URG FIN Scan"; flags: FU;)
alert tcp any any -> $HOME_NET any (msg:"PUSH FIN Scan"; flags: FP;)
alert tcp any any -> $HOME_NET any (msg:"URG PUSH Scan"; flags: PU;)
alert tcp any any -> $HOME_NET any (flags: A; ack: 0; msg:"NMAP TCP ping!";)
# IMAP buffer overflow
alert tcp any any -> $HOME_NET 143 (msg:"IMAP Buffer Overflow!"; content:"|E8
C0FF FFFF|"; flags: PA;)
# x86 named buffer overflow
alert tcp any any -> $HOME_NET 53 (msg:"named Buffer Overflow!"; content:"|CD80
E8D7 FFFF FF|"; flags: PA;)
```

To break it down, let's look at a simple rule:

```
log tcp any any -> 192.168.1.0/24 23
```

This rule says to log tcp traffic coming from any IP address and any source port to this network where the destination port is 23 (telnet).

```
log tcp any any -> any 22 (msg:”Geez! Someone’s trying to use SSH!”;)
```

Here we can see how a message can be appended to the packets that we capture. In this case we’re looking for any incoming traffic going to the standard SSH port.

Rules can also be bi-directional and look for traffic going in either direction:

```
log tcp any any <> any 23
```
This rule logs all tcp traffic where the destination port is 23 (telnet).

Messages can be added to alerts to make the alert logs a bit easier to read:

```
alert tcp any any -> 192.168.1.0/24 23 (msg:"Hey, someone's telnetting to my
network!";)
```

Resources

For more information on snort rules go to www.snort.org. They have a really cool rules database that you can get predefined rules from. There are several sites that support snort rules including whitehats.com. Enjoy!

http://www.snort.org
http://www.whitehats.com

Output Examples

Ping

```
-*> Snort! <*-
Version 1.5.1
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
Decoding Ethernet on interface ed0
11/12-22:17:48.490312 ARP who-has 192.168.1.1 tell 192.168.1.2

11/12-22:17:48.490409 ARP reply 192.168.1.1 is-at 0:20:18:B8:FE:B

11/12-22:17:48.490746 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x62
192.168.1.2 -> 192.168.1.1 ICMP TTL:64 TOS:0x0 ID:5087
ID:52519   Seq:0  ECHO
3A 0F 5E 2D 00 09 FA 2F 08 09 0A 0B 0C 0D 0E 0F  :.^-.../........
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

11/12-22:17:48.490830 0:20:18:B8:FE:B -> 0:5:2:62:E8:14 type:0x800 len:0x62
192.168.1.1 -> 192.168.1.2 ICMP TTL:255 TOS:0x0 ID:930
ID:52519   Seq:0  ECHO REPLY
3A 0F 5E 2D 00 09 FA 2F 08 09 0A 0B 0C 0D 0E 0F  :.^-.../........
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
```

Traceroute

```
-*> Snort! <*-
Version 1.5.1
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
Decoding Ethernet on interface ed0
11/12-22:18:29.612633 ARP who-has 192.168.1.1 tell 192.168.1.2

11/12-22:18:29.612733 ARP reply 192.168.1.1 is-at 0:20:18:B8:FE:B

11/12-22:18:29.613033 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x3C
192.168.1.2:42959 -> 192.168.1.1:33435 UDP TTL:1 TOS:0x0 ID:42960
Len: 20
```

```
01 01 00 00 3A 0F 5E 56 00 0B CE 8F 00 00 00 00  ....:.^V........
00 00                                            ..

11/12-22:18:29.613117 0:20:18:B8:FE:B -> 0:5:2:62:E8:14 type:0x800 len:0x46
192.168.1.1 -> 192.168.1.2 ICMP TTL:255 TOS:0x0 ID:933
DESTINATION UNREACHABLE: PORT UNREACHABLE
00 00 00 00 45 00 00 28 D0 A7 00 00 01 11 8E A1  ....E..(........
C0 A8 01 02 C0 A8 01 01 A7 CF 82 9B 00 14 00 00  ................

11/12-22:18:29.615732 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x3C
192.168.1.2:42959 -> 192.168.1.1:33436 UDP TTL:1 TOS:0x0 ID:42961
Len: 20
02 01 00 00 3A 0F 5E 56 00 0B DB 35 00 00 00 00  ....:.^V...5....
00 00                                            ..

11/12-22:18:29.615788 0:20:18:B8:FE:B -> 0:5:2:62:E8:14 type:0x800 len:0x46
192.168.1.1 -> 192.168.1.2 ICMP TTL:255 TOS:0x0 ID:934
DESTINATION UNREACHABLE: PORT UNREACHABLE
00 00 00 00 45 00 00 28 D1 A7 00 00 01 11 8E A0  ....E..(........
C0 A8 01 02 C0 A8 01 01 A7 CF 82 9C 00 14 00 00  ................

11/12-22:18:29.616745 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x3C
192.168.1.2:42959 -> 192.168.1.1:33437 UDP TTL:1 TOS:0x0 ID:42962
Len: 20
03 01 00 00 3A 0F 5E 56 00 0B DF 3C 00 00 00 00  ....:.^V...<....
00 00                                            ..

11/12-22:18:29.616798 0:20:18:B8:FE:B -> 0:5:2:62:E8:14 type:0x800 len:0x46
192.168.1.1 -> 192.168.1.2 ICMP TTL:255 TOS:0x0 ID:935
DESTINATION UNREACHABLE: PORT UNREACHABLE
00 00 00 00 45 00 00 28 D2 A7 00 00 01 11 8E 9F  ....E..(........
C0 A8 01 02 C0 A8 01 01 A7 CF 82 9D 00 14 00 00  ................
```

Telnet

```
-*> Snort! <*-
Version 1.5.1
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
Decoding Ethernet on interface ed0
11/12-22:20:09.234983 ARP who-has 192.168.1.1 tell 192.168.1.2

11/12-22:20:09.235079 ARP reply 192.168.1.1 is-at 0:20:18:B8:FE:B

11/12-22:20:09.235384 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x4A
192.168.1.2:1176 -> 192.168.1.1:23 TCP TTL:64 TOS:0x0 ID:5235  DF
S***** Seq: 0x6FFFFDB7   Ack: 0x0   Win: 0x7D78
TCP Options => MSS: 1460 SackOK TS: 649836 0 NOP WS: 0

11/12-22:20:09.235525 0:20:18:B8:FE:B -> 0:5:2:62:E8:14 type:0x800 len:0x3A
192.168.1.1:23 -> 192.168.1.2:1176 TCP TTL:64 TOS:0x0 ID:1027  DF
S***A* Seq: 0x62E9DFFD   Ack: 0x6FFFFDB8   Win: 0x4470
TCP Options => MSS: 1460

…
…

11/12-22:20:09.334644 0:20:18:B8:FE:B -> 0:5:2:62:E8:14 type:0x800 len:0x6A
```

```
192.168.1.1:23 -> 192.168.1.2:1176 TCP TTL:64 TOS:0x10 ID:1036  DF
***PA* Seq: 0x62E9E073   Ack: 0x6FFFFE47   Win: 0x4470
0D 00 0D 0A 46 72 65 65 42 53 44 2F 69 33 38 36  ....FreeBSD/i386
20 28 62 73 64 6D 61 63 68 69 6E 65 29 20 28 74   (bsdmachine) (t
74 79 70 30 29 0D 00 0D 0A 0D 00 0D 0A 6C 6F 67  typ0)........log
69 6E 3A 20                                      in:

11/12-22:20:09.354517 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x3C
192.168.1.2:1176 -> 192.168.1.1:23 TCP TTL:64 TOS:0x0 ID:5246  DF
****A* Seq: 0x6FFFFE47   Ack: 0x62E9E0A7   Win: 0x7D78
00 00 00 00 00 00                                ......



11/12-22:20:10.388357 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x3C
192.168.1.2:1176 -> 192.168.1.1:23 TCP TTL:64 TOS:0x0 ID:5248  DF
***PA* Seq: 0x6FFFFE47   Ack: 0x62E9E0A7   Win: 0x7D78
6B 00 00 00 00 00                                k.....

11/12-22:20:10.388619 0:20:18:B8:FE:B -> 0:5:2:62:E8:14 type:0x800 len:0x37
192.168.1.1:23 -> 192.168.1.2:1176 TCP TTL:64 TOS:0x10 ID:1037  DF
***PA* Seq: 0x62E9E0A7   Ack: 0x6FFFFE48   Win: 0x4470
6B                                               k

11/12-22:20:10.404587 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x3C
192.168.1.2:1176 -> 192.168.1.1:23 TCP TTL:64 TOS:0x0 ID:5249  DF
****A* Seq: 0x6FFFFE48   Ack: 0x62E9E0A8   Win: 0x7D78
00 00 00 00 00 00                                ......

…
…

11/12-22:20:11.024771 0:20:18:B8:FE:B -> 0:5:2:62:E8:14 type:0x800 len:0x3F
192.168.1.1:23 -> 192.168.1.2:1176 TCP TTL:64 TOS:0x10 ID:1043  DF
***PA* Seq: 0x62E9E0AE   Ack: 0x6FFFFE4E   Win: 0x4470
50 61 73 73 77 6F 72 64 3A                       Password:

11/12-22:20:11.044621 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x3C
192.168.1.2:1176 -> 192.168.1.1:23 TCP TTL:64 TOS:0x0 ID:5260  DF
****A* Seq: 0x6FFFFE4E   Ack: 0x62E9E0B7   Win: 0x7D78
00 00 00 00 00 00                                ......

11/12-22:20:11.697108 0:5:2:62:E8:14 -> 0:20:18:B8:FE:B type:0x800 len:0x3C
192.168.1.2:1176 -> 192.168.1.1:23 TCP TTL:64 TOS:0x0 ID:5261  DF
***PA* Seq: 0x6FFFFE4E   Ack: 0x62E9E0B7   Win: 0x7D78
56 00 00 00 00 00                                V.....
```