CHAPTER 19

Snort

Introduction

Snort is an open source network intrusion detection system that can be installed on Linux and Windows. It functions by first normalizing traffic, then checking the traffic against sets of rules. There are community rules, registered rules, and commercial rules for Snort available from http://www.snort.org; it is also possible to write custom rules. To avoid false positives, Snort needs to be tuned for its environment. Snort can raise alerts when specific traffic is seen on the network; it can also detect port scans, ARP spoofing, and sensitive data like credit card numbers or social security numbers.

Installing Snort

Snort is available for both Linux and Windows.

Installing Snort on Linux

Binary packages are available for Snort for some distributions, including CentOS 7, but they are not available for all the distributions under consideration. Snort can be compiled from source on all the distributions under consideration.

Downloading Snort

To compile Snort from source, two packages must be downloaded and compiled: daq, which performs data acquisition, and snort, which performs intrusion detection. These are available from https://snort.org/downloads, though usually only the most recent versions are available. As of this writing, the most recent version of daq is 2.0.6, while the most recent version of snort is 2.9.11.1. Snort 3.0 is under development but is still an alpha release.

M. O'Leary, *Cyber Operations*, https://doi.org/10.1007/978-1-4842-4294-0_19

To proceed, first download the archives.[1]

```
nunki:~ # wget https://snort.org/downloads/snort/daq-2.0.6.tar.gz
nunki:~ # wget https://snort.org/downloads/snort/snort-2.9.11.1.tar.gz
```

These can then be uncompressed in a convenient directory, say /usr/local/src.

```
nunki:~ # tar -xzvf ./daq-2.0.6.tar.gz -C /usr/local/src
nunki:~ # tar -xzvf ./snort-2.9.11.1.tar.gz -C /usr/local/src
nunki:~ # ls -l /usr/local/src/
total 8
drwxr-xr-x  6 cgauss 1000 4096 Jul  8 10:57 daq-2.0.6
drwxr-xr-x 10 root   root 4096 Jul  9 22:54 snort-2.9.11.1
```

Second, the target system requires a collection of supporting packages to be present. These include:

- gcc
- make
- flex
- bison
- pcap development libraries
- pcre development libraries
- dnet development libraries
- zlib development libraries
- lzma development libraries

The process to install these packages and libraries varies with the distribution.

Installing Snort Dependencies on OpenSuSE

To install the needed Snort dependencies on OpenSuSE, the administrator can run the command

```
nunki:~ # zypper install gcc make flex bison libpcap-devel pcre-devel libdnet-devel
zlib-devel xz-devel
```

[1]Be sure to verify the MD5 sums!

Installing Snort Dependencies on CentOS 6, 7

To install the needed Snort dependencies on CentOS 6 or CentOS 7, begin by enabling the EPEL repository; this contains the needed dnet development libraries.[2]

The needed packages can be installed with the following command.

```
[root@kakkab ~]# yum install gcc make flex bison libpcap-devel pcre-devel libdnet-devel
zlib-devel xz-devel
```

Installing Snort Dependencies on CentOS 5

The situation for CentOS 5 is more complex. CentOS 5 provides older versions of libpcap; for example, CentOS 5.9 uses libpcap 0.9.4. However, Snort needs at least version 1.0.0.

To proceed, first install the same packages as CentOS 6 or 7, but without libpcap. This requires EPEL for the dnet development libraries.

```
[root@alnitak ~]# yum install gcc make flex bison pcre-devel libdnet-devel zlib-devel
xz-devel
```

Next, any existing copies of libpcap should be removed. This may result in removal of other installed software that requires libpcap.

```
[root@alnitak src]# yum remove libpcap libpcap-devel
```

Next, download libpcap 1.7.4 from http://www.tcpdump.org. This should be uncompressed in a convenient location, say /usr/local/src.

```
[root@alnitak ~]# wget http://www.tcpdump.org/release/libpcap-1.7.4.tar.gz
[root@alnitak ~]# tar -xzvf ./libpcap-1.7.4.tar.gz -C /usr/local/src
```

Compile libpcap from source using configure, make, and make install. Because this is going to replace the system file, it is installed in /usr rather than in /usr/local. When the installation is complete, ldconfig needs to be run to update the system with the location of the libraries.

```
[root@alnitak ~]# cd /usr/local/src/libpcap-1.7.4
[root@alnitak libpcap-1.7.4]# ./configure --prefix=/usr
[root@alnitak libpcap-1.7.4]# make
[root@alnitak libpcap-1.7.4]# make install
[root@alnitak libpcap-1.7.4]# ldconfig
```

[2]Instructions on how to enable the EPEL repository are provided in the Notes and References section of Chapter 14.

Installing Snort Dependencies on Mint or Ubuntu

On Mint or Ubuntu systems, the required packages can be installed by running

```
cgauss@eskimo ~ $ sudo apt install gcc make flex bison libpcap-dev libpcre3-dev
libdumbnet-dev zlib1g-dev liblzma-dev
```

Note that the names of the dependencies are different from those installed on other distributions. Ubuntu and Mint systems have a package named `libdnet-dev`; however this package provides libraries and headers for Linux DECnet; this is not what is needed to compile Snort. Instead, the required package is `libdumbnet`. For example, on Mint 15 the following information is provided.

```
cgauss@eskimo ~ $ apt show libdumbnet-dev
Package: libdumbnet-dev
New: yes
State: installed
Automatically installed: no
Version: 1.12-3.1
Priority: optional
Section: universe/libdevel
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Uncompressed Size: 243 k
Depends: libc6 (>= 2.7), libdumbnet1 (= 1.12-3.1)
Conflicts: libdumbnet-dev
Description: A dumb, portable networking library -- development files
 libdumbnet provides a simplified, portable interface to several low-level
 networking routines, including
 * network address manipulation
 * kernel arp(4) cache and route(4) table lookup and manipulation
 * network firewalling (Ip filter, ipfw, ipchains, pdf, ...)
 * network interface lookup and manipulation
 * raw IP packet and Ethernet frame transmission

 libdumbnet is known as libdnet in other distributions, but was renamed in
 Debian in favour of the older DECnet library 'libdnet'.

 This package contains the static library and the C header files.
Homepage: http://code.google.com/p/libdnet/
```

Compiling and Installing Snort

Once the required packages are present on the system, the administrator compiles daq by running `configure`, `make`, and `make install`.

```
nunki:~ # cd /usr/local/src/daq-2.0.6/
nunki:/usr/local/src/daq-2.0.6 # ./configure
nunki:/usr/local/src/daq-2.0.6 # make
nunki:/usr/local/src/daq-2.0.6 # make install
```

Repeat the process to configure and compile snort.

```
nunki:/usr/local/src/daq-2.0.6 # cd /usr/local/src/snort-2.9.11.1/
nunki:/usr/local/src/snort-2.9.11.1 # ./configure
nunki:/usr/local/src/snort-2.9.11.1 # make
nunki:/usr/local/src/snort-2.9.11.1 # make install
```

On some distributions, one additional step is required; the system needs to be made aware of the locations of the newly created library files. If this step is skipped, attempts to start Snort return an error of the following form.

```
nunki:/usr/local/src/snort-2.9.11.1 # snort
snort: error while loading shared libraries: libsfbpf.so.0: cannot open shared
object file: No such file or directory
```

Run the following command.

```
nunki:/usr/local/src/snort-2.9.11.1 # ldconfig
```

Then Snort should start normally.

Installing Snort on Windows

Snort can be installed on Windows; binaries including a Windows installer are available from the Snort download page at https://snort.org/downloads. Snort on Windows requires WinPcap; that program is available from https://www.winpcap.org/install/.

Snort as a Packet Sniffer

Once Snort is installed, running it from the command line starts it as a packet sniffer. In this mode Snort prints observed packet headers to the screen; the process can be stopped with CTRL+C.

```
C:\Users\jbach>c:\Snort\bin\snort.exe
Running in packet dump mode
```

```
        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{5463A773-3A5E-4F2E-A93D-
4FD3592B2D94}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.11.1-WIN32 GRE (Build 268)
   ''''  By Martin Roesch & The Snort Team:
                  http://www.snort.org/contact#team
           Copyright (C) 2014-2017 Cisco and/or its affiliates. All rights
                  reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.38 2015-11-23
           Using ZLIB version: 1.2.3

Commencing packet processing (pid=2668)
07/08-14:56:25.631178 10.0.15.204:61305 -> 10.0.15.200:53
UDP TTL:128 TOS:0x0 ID:25651 IpLen:20 DgmLen:79
Len: 51
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
07/08-14:56:25.631514 10.0.15.200:53 -> 10.0.15.204:61305
UDP TTL:128 TOS:0x0 ID:14684 IpLen:20 DgmLen:189
Len: 161
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/08-14:56:25.633357 10.0.15.204:58318 -> 65.55.44.109:443
TCP TTL:128 TOS:0x0 ID:23101 IpLen:20 DgmLen:52 DF
******S* Seq: 0xC06068F0  Ack: 0x0  Win: 0x2000  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 8 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
07/08-14:56:25.656432 65.55.44.109:443 -> 10.0.15.204:58318
TCP TTL:255 TOS:0x0 ID:40738 IpLen:20 DgmLen:44
***A**S* Seq: 0x10A2793  Ack: 0xC06068F1  Win: 0x8000  TcpLen: 24
TCP Options (1) => MSS: 1460
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
... Output Deleted ...
*** Caught Int-Signal
===============================================================================
Run time for packet processing was 26.985000 seconds
Snort processed 108 packets.
Snort ran for 0 days 0 hours 0 minutes 26 seconds
   Pkts/sec:            4
===============================================================================
Packet I/O Totals:
   Received:          108
   Analyzed:          108 (100.000%)
    Dropped:            0 (  0.000%)
   Filtered:            0 (  0.000%)
Outstanding:            0 (  0.000%)
   Injected:            0
===============================================================================
Breakdown by protocol (includes rebuilt packets):
        Eth:          108 (100.000%)
       VLAN:            0 (  0.000%)
        IP4:          104 ( 96.296%)
       Frag:            0 (  0.000%)
       ICMP:            0 (  0.000%)
        UDP:           40 ( 37.037%)
        TCP:           64 ( 59.259%)
        IP6:            0 (  0.000%)
    IP6 Ext:            0 (  0.000%)
... Output Deleted ...
       MPLS:            0 (  0.000%)
        ARP:            4 (  3.704%)
... Output Deleted ...
     S5 G 2:            0 (  0.000%)
      Total:          108
===============================================================================
Snort exiting
```

If Snort is run with the -e flag, information about the link-layer is shown.

```
cgauss@Hispania:~$ sudo snort -e
Running in packet dump mode
```

```
        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "enp0s3".
Decoding Ethernet

... Output Deleted ...

07/08-18:01:09.706032 08:00:27:25:09:5B -> 52:54:00:12:35:00 type:0x800 len:0x36
10.0.0.76:52674 -> 23.217.129.144:80 TCP TTL:64 TOS:0x0 ID:33322 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xEAE344F7  Ack: 0x11E95A8  Win: 0x88E0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/08-18:01:09.706184 08:00:27:25:09:5B -> 52:54:00:12:35:00 type:0x800 len:0x36
10.0.0.76:46652 -> 35.227.212.213:80 TCP TTL:64 TOS:0x0 ID:12645 IpLen:20
DgmLen:40 DF
***A**** Seq: 0xE2EBF267  Ack: 0x11BE934  Win: 0x8610  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

... Output Deleted ...
```

If the -d flag is passed, then Snort displays the full packet content.

```
dschubba:~ # snort -d
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

... Output Deleted ...

WARNING: No preprocessors configured for policy 0.
07/08-18:05:21.175335 52.203.89.231:9571 -> 10.0.0.76:58246
TCP TTL:255 TOS:0x0 ID:53597 IpLen:20 DgmLen:434
***AP*** Seq: 0x144CEF0  Ack: 0xFA8BA630  Win: 0x7B7C  TcpLen: 20
81 7E 01 86 7B 22 70 6C 22 3A 22 7B 5C 22 74 73  .~..{"pl":"{\"ts
5C 22 3A 31 35 33 31 30 38 37 35 32 31 31 35 31  \":1531087521151
2C 5C 22 7E 63 5C 22 3A 31 2C 5C 22 70 6C 5C 22  ,\"~c\":1,\"pl\"
3A 5C 22 65 4A 79 31 6B 46 30 4C 67 6A 41 55 68  :\"eJy1kF0LgjAUh
76 2F 4B 32 4C 58 67 56 4C 4C 30 4D 71 4C 62 6F  v/K2LXgVLL0MqLbo
4C 71 4C 4C 70 59 65 55 5A 68 75 62 47 65 52 52  LqLLpYeUZhubGeRR
50 37 32 52 6B 46 66 52 47 44 5A 33 58 6A 48 65  P72RkFfRGDZ3XjHe
```

```
5A 37 33 6E 4D 32 52 53 6B 56 54 71 6B 45 4A 6E  Z73nM2RSkVTqkEJn
67 48 31 71 4F 4A 59 75 73 43 6B 51 53 66 53 67  gH1qOJYusCkQSfSg
48 57 51 52 68 4D 32 5A 70 4D 67 6A 4C 76 73 2F  HWQRhM2ZpMgjLvs/
76 5A 33 33 4D 44 53 4E 67 31 6F 73 34 59 44 75  vZ33MDSNg1os4YDu
73 6B 39 46 78 62 63 36 44 55 6C 73 69 46 46 70  sk9Fxbc6DUlsiFFp
51 33 53 6B 2F 65 31 52 44 62 7A 43 2B 49 47 6A  Q3Sk/e1RDbzC+IGj
38 4A 52 7A 48 6F 42 6B 77 64 67 38 71 48 31 31  8JRzHoBkwdg8qH11
50 30 59 41 72 58 43 74 6C 2F 6A 5A 30 46 68 68  P0YArXCtl/jZ0Fhh
56 67 68 52 32 74 38 42 62 71 53 2B 56 30 52 44  VghR2t8BbqS+V0RD
34 50 46 56 6F 47 66 41 2F 4A 4B 50 4E 52 66 53  4PFVoGfA/JKPNRfS
30 56 69 78 78 72 51 59 55 71 70 63 66 59 48 6B  0VixxrQYUqpcfYHk
62 54 34 65 6E 31 47 46 69 37 38 42 54 72 6B 74  bT4en1GFi78BTrkt
59 32 74 61 36 37 62 64 31 74 76 7A 37 37 6A 2B  Y2ta67bd1tvz77j+
4C 6B 3D 5C 22 7D 22 2C 22 6F 70 22 3A 22 50 22  Lk=\"}","op":"P"
2C 22 74 63 22 3A 22 73 63 6F 72 65 73 2D 65 73  ,"tc":"scores-es
70 6E 2D 65 6E 2D 66 72 6F 6E 74 70 61 67 65 2D  pn-en-frontpage-
69 6E 64 65 78 2D 65 6E 2D 75 73 22 2C 22 6D 69  index-en-us","mi
64 22 3A 39 38 34 32 35 31 7D                    d":984251}

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The -v flag can be passed to provide verbose output.

Snort can be used like tcpdump or Wireshark and store the contents of the sniffed traffic in a binary file. Consider the sequence

```
dschubba:~ # mkdir captures
dschubba:~ # snort -l ./captures/
```

This tells Snort to sniff packets and store the result in the directory ~/captures (which must already exist). If Snort is run for a time, and the directory examined, the contents will be like the following.

```
dschubba:~ # ls -l ./captures/
total 212
-rw------- 1 root root 215203 Jul  8 19:10 snort.log.1531091394
```

Here the file name is snort.log.1531091394 where the number 1531091394 is the Unix timestamp of the date/time the packet capture was made.

```
dschubba:~ # date -d @1531091394
Sun Jul  8 19:09:54 EDT 2018
```

The resulting file can then be opened in tools like Wireshark or tcpdump.

Snort as an Intrusion Detection System

Although Snort can be run as a packet sniffer, its purpose is to act as an intrusion detection system (IDS).

Rule Installation

To use Snort as an IDS, it must be provided with a rule set; there are three rule sets available from https://www.snort.org/downloads. The community rule set is developed from user submissions, and it is freely available and released under the GPL (v2). The subscriber rule set is available for purchase and contains the most recent rules. The registered rule set is available without a fee for users who register; it is based on subscriber rules that are at least 30 days old and includes the community rules.

Start by downloading the rule set, creating the directory `/etc/snort`, and unpacking the rule set there.

```
jmaxwell@diomedes:~$ wget https://www.snort.org/downloads/registered/snortrules-
snapshot-29111.tar.gz
jmaxwell@diomedes:~$ sudo mkdir /etc/snort
jmaxwell@diomedes:~$ sudo tar -xzvf ./snortrules-snapshot-29111.tar.gz
-C /etc/snort/
```

This provides four subdirectories.

```
jmaxwell@diomedes:~$ ls -F /etc/snort/
etc/  preproc_rules/  rules/  so_rules/
```

The `/etc/snort/etc` subdirectory contains configuration information. The remaining three directories contain rules. Many of the rules are provided in a plaintext format; this makes them easy to read and modify.

Installing Precompiled Rules

Some rules are provided as precompiled shared objects (`.so` files). There are different versions of the binary rules, depending on the distribution and architecture.

```
jmaxwell@diomedes:~$ ls -F /etc/snort/so_rules/precompiled/
Centos-5-4/  FC-25/         FreeBSD-9-0/    RHEL-5-5/        Ubuntu-14-4/
Centos-6/    FC-26/         OpenBSD-5-2/    RHEL-6/          Ubuntu-16-4/
Centos-7/    FC-27/         OpenBSD-5-3/    RHEL-6-0/        Ubuntu-17-10/
Debian-7/    FreeBSD-10-0/  OpenBSD-6-2/    RHEL-7/
Debian-8/    FreeBSD-11/    OpenSUSE-15-0/  Slackware-13-1/
Debian-9/    FreeBSD-8-1/   OpenSUSE-42-3/  Slackware-14-2/
```

Most directories contain a pair of subdirectories for 32-bit and 64-bit architectures, and these contain a subdirectory that matches the rule version. For example:[3]

```
jmaxwell@diomedes:~$ tree -d /etc/snort/so_rules/precompiled/Ubuntu-16-4/
/etc/snort/so_rules/precompiled/Ubuntu-16-4/
├── i386
│   └── 2.9.11.1
└── x86-64
    └── 2.9.11.1
```

To prepare the system to use these dynamic rules, a link must be created. Suppose that Snort has been installed on a 64-bit Ubuntu 16.10 system. Then a natural choice for the dynamic rules would be the 64-bit Ubuntu 16.04 rules. Create the symbolic link to `/usr/local/lib/snort_dynamicrules`.

```
jmaxwell@diomedes:~$ sudo ln -s /etc/snort/so_rules/precompiled/Ubuntu-16-4/x86-64/2.9.11.1/ /usr/local/lib/snort_dynamicrules
```

Snort Reputation Preprocessor

The default Snort configuration file `/etc/snort/etc/snort.conf` configures a reputation preprocessor that uses a pair of files for its whitelist and blacklist; these files need to be created.

```
jmaxwell@diomedes:~$ sudo touch /etc/snort/rules/white_list.rules
jmaxwell@diomedes:~$ sudo touch /etc/snort/rules/black_list.rules
```

Snort Log Directory

With its default settings, Snort stores its results in the log directory `/var/log/snort`, which must exist.

```
jmaxwell@diomedes:~$ sudo mkdir /var/log/snort
```

Starting Snort as an Intrusion Detection System

With this last change made, Snort can be started as an IDS using the default configuration file by running

```
jkepler@Coperniucs:~$ sudo snort -c /etc/snort/etc/snort.conf
```

[3]The `tree` command is used to show files and directories in a tree structure. The `-d` flag restricts the output to directories only. This is not part of a typical Linux installation, but can be installed on, for example, Ubuntu with `apt install tree`.

Running Snort as an IDS on CentOS 5

Attempts to start Snort as an IDS on CentOS 5 may fail with the error

```
[root@alnair ~]# snort -c /etc/snort/etc/snort.conf
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/etc/snort.conf"

... Output Deleted ...

++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
ERROR: /etc/snort/etc/../rules/browser-ie.rules(380) : pcre compile of
"<area\s[^>]*?id\s*?=\s*?[\x22\x27]?(?P<area_id>[^\x22\x27\s>]+)[\x22\x27]?
[\s>].*?(?P=area_id)\.attachEvent\s*?\x28[^\x29]+?,[^\x29]*?(?<func>[^\x29\s]+)
[\s\x29].*?(?P=func)[^\x7b]+?\x7b[^\x7d]*?(?P=area_id)\.detachEvent\s*?\x28
[^\x7d]+?document\.open" failed at offset 137 : unrecognized character after (?<
Fatal Error, Quitting..
```

The issue here is that the CentOS 5 PCRE package cannot compile this rule. Further, this PCRE package is used extensively throughout CentOS 5 and is difficult to replace; even yum depends on it. This problem only occurs in one rule set started by default, namely the rules for Internet Explorer. If these rules are omitted, Snort starts.

Running Snort as an IDS on OpenSuSE

One potential problem running Snort as an IDS on 64-bit versions of OpenSuSE 42 is that the compilation process installs key Snort libraries to `/usr/local/lib64` rather than `/usr/local/lib`. One solution is to modify the configuration file `/etc/snort/etc/snort.conf`. Modify the locations for the dynamic preprocessor libraries and the base preprocessor engine as follows in Listing 19-1.

Listing 19-1. Portion of the file `/etc/snort/etc/snort.conf` on OpenSuSE 42.2 with modified locations for the dynamic preprocessor libraries and the base preprocessor engine

```
##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
##################################################
```

```
# path to dynamic preprocessor libraries
dynamicpreprocessor directory /usr/local/lib64/snort_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine /usr/local/lib64/snort_dynamicengine/libsf_engine.so

# path to dynamic rules libraries
dynamicdetection directory /usr/local/lib/snort_dynamicrules
```

This section also contains the location of the dynamic rules libraries. Rather than follow the instructions earlier that created a symbolic link from `/usr/local/lib/snort_dynamicrules` to the proper subdirectory of `/etc/snort`, the administrator could instead modify the path to the dynamic rules libraries here to point to the proper location.

Another problem is that Snort only includes precompiled rules for OpenSuSE 42.3 and OpenSuSE 15.0 and both sets of precompiled rules are for 64-bit systems only.[4]

Further, older OpenSuSE systems use older versions of glibc; for example, OpenSuSE 11.4 includes glibc 2.11.3 by default. On such a 64-bit system, attempts to start Snort as an IDS with the current rule set fails.

```
diphda:~ # snort -c /etc/snort/etc/snort.conf
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/etc/snort.conf"

... Output Deleted ...

Loading dynamic engine /usr/local/lib/snort_dynamicengine/libsf_engine.so... done
Loading all dynamic detection libs from /usr/local/lib/snort_dynamicrules...
  Loading dynamic detection library /usr/local/lib/snort_dynamicrules/
  file-executable.so... done
  Loading dynamic detection library /usr/local/lib/snort_dynamicrules/
  os-linux.so... done
  Loading dynamic detection library /usr/local/lib/snort_dynamicrules/server-
  other.so... ERROR: Failed to load /usr/local/lib/snort_dynamicrules/server-
  other.so: /lib64/libc.so.6: version `GLIBC_2.14' not found (required by /usr/
  local/lib/snort_dynamicrules/server-other.so)
Fatal Error, Quitting..
```

[4]OpenSuSE 15.0 was released in May 2018, and it follows after OpenSuSE 42.3. The means that OpenSuSE major releases come in the following order: 12 ➤ 13 ➤ 42 ➤ 15. I am not sure why this is confusing to anyone.

Running Snort as an IDS on Mint or Ubuntu

Most versions of Mint and Ubuntu can use the provided instructions to run Snort as an IDS. However, older versions of Mint, including Mint 11 and Mint 12, use version 2.13 for glibc, but the Snort rules expect version 2.14 or later.

Running Snort as an IDS on Windows

Snort can be configured on a Windows system to run as an IDS. Download a copy of the rule set and uncompress the result. Move the subdirectories `rules` and `preproc_rules` to the (default) directory `C:\Snort\rules`. The rule set also contains the directory `so_rules`; this consists of various rules shipped in binary form; however, these do not run on Windows. (See the file `so_rules\src\README` for details.) The contents of the `etc` directory in the rule set are copied to `C:\Snort\etc`.

The configuration file `C:\Snort\etc\snort.conf` is modified with the correct paths for various configuration files. Update the location of the dynamic preprocessor libraries and engine with the corresponding values on a Windows installation and comment out the dynamic rules libraries.

```
# path to dynamic preprocessor libraries
dynamicpreprocessor directory C:\Snort\lib\snort_dynamicpreprocessor

# path to base preprocessor engine
dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll

# path to dynamic rules libraries
#dynamicdetection directory /usr/local/lib/snort_dynamicrules
```

The location of the rules files is also updated.

```
# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH C:\Snort\rules\rules
var SO_RULE_PATH C:\Snort\rules\so_rules
var PREPROC_RULE_PATH C:\Snort\rules\preproc_rules

var WHITE_LIST_PATH  C:\Snort\rules\rules
var BLACK_LIST_PATH  C:\Snort\rules\rules
```

The files `C:\Snort\rules\rules\white_list.rules` and `C:\Snort\rules\rules\black_list.rules` must exist, though they can be blank.

Snort can decompress LZMA files, but only if support is included when the binary is compiled. This is not the case for the default Windows binary. However, the default Snort configuration file includes LZMA as one of the methods it can apply for `.swf` files, so attempts to start Snort as an IDS on Windows will fail with the error

```
c:\Snort>c:\Snort\bin\snort.exe -c c:\Snort\etc\snort.conf
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!

... Output Deleted ...

ERROR: c:\Snort\etc\snort.conf(326) => Invalid keyword '}' for server
configuration.
Fatal Error, Quitting..
```

To resolve the error, the corresponding line (326) of the Snort configuration file needs to be modified. Modify the line

```
decompress_swf { deflate lzma } \
```

Instead use the line

```
decompress_swf { deflate } \
```

Once these changes are made, Snort can be started from within the Snort directory[5] as an IDS with the command

```
c:\Snort>c:\Snort\bin\snort.exe -c c:\Snort\etc\snort.conf
```

Testing Snort

Once Snort can start without errors, either on Windows or on Linux, the next step is to verify that is it correctly seeing traffic and responding with alerts.

Creating Custom Snort Rules

One approach to validating the install is to craft a Snort testing rule that fires on specified traffic. The file `/etc/snort/rules/local.rules` (`C:\Snort\rules\rules\local.rules` on Windows) is designed to contain rules that are local to a sensor. To that file, add the testing rule

```
alert tcp any any <> any any (content:"shibboleth"; nocase; msg:"Snort Shibboleth
Testing Rule"; sid:1000001; rev:1)
```

[5]By default, Snort uses a relative directory (`..\log\alert.ids`) to store any alerts; if this directory does not exist, Snort fails to start. This can also be avoided by specifying the absolute path for the log file, by running `c:\>c:\Snort\bin\snort.exe -c c:\Snort\etc\snort.conf -l C:\Snort\log`

This rule generates an alert whenever Snort observes TCP traffic traveling between arbitrary addresses and arbitrary ports that contains the text "shibboleth," regardless of the case of the text. With this rule in place, restart the Snort sensor and visit such a web page containing the word "shibboleth." If Snort is functioning correctly, then the Snort alert file `/var/log/snort/alert` (`C:\Snort\log\alert.ids` on Windows) shows alerts like

```
[**] [1:1000001:1] Snort Shibboleth Testing Rule [**]
[Priority: 0]
07/29-06:06:03.393808 10.0.2.28:80 -> 10.0.15.204:55741
TCP TTL:64 TOS:0x0 ID:46159 IpLen:20 DgmLen:504 DF
***AP*** Seq: 0x57C8A69  Ack: 0xA5C607B9  Win: 0x36  TcpLen: 20
```

When using this testing the rule, be aware that modern browsers generally announce that they will accept compressed responses. Here is a typical request/response pair from Firefox 52.9 to Apache on Ubuntu 14.04:

```
GET /index.html HTTP/1.1
Host: 10.0.3.48
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Thu, 02 Aug 2018 01:05:27 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Thu, 02 Aug 2018 01:05:10 GMT
ETag: "d9-57269671c6841-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 162
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

...............O.D.|..O..@.......1mLc...1..{R"f$..Y.{.........u....A>8.Xe........
wY........[...fm..6......v.....cs../..E...........c.%`.|..R....4........>.[......
```

The bolded portion of this listing shows how the browser announced that it would accept gzip encoded data, and that the provided content was gzip encoded. This same behavior is observed in Figure 14-2. The Snort testing rule would not fire on this traffic as the required shibboleth is not present.

In other cases, the browser may have a cached copy of the page, and so makes a conditional request.[6] The server may then respond with HTTP/304 Not Modified. This response also does not include the required shibboleth.

To test Snort rules, an administrator can use the techniques of Chapter 14 to make manual requests. Other options include using Linux commands like wget to download the web page, or to configure the server not to compress pages even when requested. On Apache this is handled by mod_deflate.[7] On IIS, select Compression from IIS Manager.

Snort and Packet Captures

Snort can read and process alerts from a file rather than directly from a network interface using the `-r` flag. To process the packet capture file `data.pcap` with the configuration file `/etc/snort/etc/snort.conf`, run

```
[root@scheat ~]# snort -r ./data.pcap -c /etc/snort/etc/snort.conf
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/etc/snort.conf"

... Output Deleted ...

Commencing packet processing (pid=3761)
===============================================================================
Run time for packet processing was 0.12787 seconds
Snort processed 32 packets.
Snort ran for 0 days 0 hours 0 minutes 0 seconds
   Pkts/sec:           32
```

[6] https://tools.ietf.org/html/rfc7232, https://developer.mozilla.org/en-US/docs/Web/HTTP/Conditional_requests

[7] https://httpd.apache.org/docs/2.4/mod/mod_deflate.html, https://httpd.apache.org/docs/2.2/mod/mod_deflate.html

```
... Output Deleted ...

===============================================================================
Action Stats:
     Alerts:            1 (  2.941%)
     Logged:            1 (  2.941%)
     Passed:            0 (  0.000%)
```

Running Snort against a known packet capture is an excellent way to debug rules and the configuration files.

Running Snort as a Service

To be most useful as an IDS, Snort should start automatically and run as a service under a separate (non-root) user.

Snort Users and Permissions

The first step on Linux is to create an unprivileged user and an unprivileged group to run Snort. For example, on CentOS 6.8, an administrator can run the following.

```
[root@scheat ~]# groupadd snort
[root@scheat ~]# useradd -r -g snort -s /sbin/nologin snort
```

The first command creates the group snort, the second creates the user snort as a system account (`-r`) in the group snort (`-g snort`) and disables the login shell (`-s /sbin/nologin`). The location of the disabled logon shell may vary; Ubuntu and Mint use `/usr/sbin/nologin`.

During the installation process, the directory `/var/log/snort` was created; it should exist with the proper permissions:

```
[root@scheat ~]# chown snort:snort /var/log/snort
[root@scheat ~]# ls -ld /var/log/snort
drwxr-xr-x. 2 snort snort 4096 Jul 29 09:22 /var/log/snort
```

By default, Snort stores alerts in the file `/var/log/snort/alert`; ensure that this file exists and has the proper permissions.

```
[root@scheat ~]# touch /var/log/snort/alert
[root@scheat ~]# chown snort:snort /var/log/snort/alert
[root@scheat ~]# chmod 600 /var/log/snort/alert
[root@scheat ~]# ls -l /var/log/snort/alert
-rw-------. 1 snort snort 1430 Jul 29 09:22 /var/log/snort/alert
```

Configuring Snort as a Service on CentOS 5/6

The process to configure Snort to start as a service varies with the distribution and the method the distribution uses to manage services (SysVInit, Upstart, systemd).

CentOS 5 and 6 use SysVInit to manage services. The Snort source code includes a sample startup script that can be used to start the service. Copy the script to `/etc/init.d/` and set it as executable.

```
[root@scheat ~]# cp /usr/local/src/snort-2.9.11.1/rpm/snortd /etc/init.d/
[root@scheat ~]# chmod a+x /etc/init.d/snortd
```

A check of the content of that script shows that it calls Snort from `/usr/sbin/snort`; however, the installation process presented in this chapter stores the Snort executable in `/usr/local/bin/snort`. One solution is to create a symlink

```
[root@scheat ~]# ln -s /usr/local/bin/snort /usr/sbin/snort
```

The default startup script loads configuration data from the file `/etc/sysconfig/snort`; the source package contains a template for that file as well that can be copied into place.

```
[root@scheat ~]# cp /usr/local/src/snort-2.9.11.1/rpm/snort.sysconfig
/etc/sysconfig/snort
```

This template sets the snort configuration file to `/etc/snort/snort.conf`; update the file[8] to point to `/etc/snort/etc/snort.conf`

```
# Where is Snort's configuration file?
# -c {/path/to/snort.conf}
CONF=/etc/snort/etc/snort.conf
```

Configure Snort as a service with

```
[root@scheat ~]# chkconfig --add snortd
```

Snort can then be controlled with the service command

```
[root@scheat ~]# service snortd start
Starting snort: Spawning daemon child...
My daemon child 4379 lives...
Daemon parent exiting (0)
```

The Snort source also includes the file `/usr/local/src/snort-2.9.11.1/rpm/snort.logrotate` with the content

[8] A reasonable alternative is to store the configuration file in `/etc/snort/snort.conf`; however, this requires a change in `snort.conf`, which uses the relative path `../rules` for the location of the rules.

```
[root@scheat ~]# cat /usr/local/src/snort-2.9.11.1/rpm/snort.logrotate
# /etc/logrotate.d/snort
# $Id$

/var/log/snort/alert /var/log/snort/*log /var/log/snort/*/alert /var/log/
snort/*/*log  {
    daily
    rotate 7
    missingok
    compress
    sharedscripts
    postrotate
        /etc/init.d/snortd restart 1>/dev/null || true
    endscript
}
```

This can be copied to `/etc/logrotate.d/` to control how the system rotates the Snort logs; see Chapter 10.

Configuring Snort as a Service on CentOS 7

CentOS 7 uses systemd to manage services. However, as explained by the file `/etc/init.d/README`, traditional SysVInit scripts can be used. Consequently, the approach used for CentOS 5/6 can be used on CentOS 7, with one change. Because CentOS 7 uses `enp0s3` rather than `eth0` for the name of the primary ethernet interface, the file copied to `/etc/sysconfig/snort` must be modified with the correct name of the interface.

Another approach is to create a native systemd service file. To do so, create the file `/lib/systemd/system/snort.service` (Listing 19-2) with the following content.

Listing 19-2. The file `/lib/systemd/system/snort.service` to control Snort on CentOS 7

```
[Unit]
Description=Snort Intrusion Detection System
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/snort -A fast -b -d -D -i enp0s3 -u snort -g snort
-c /etc/snort/etc/snort.conf -l /var/log/snort

[Install]
WantedBy=multi-user.target
```

The service definition includes the full command to start Snort. This is the same command line that is used to start Snort in the SysVInit script.

The service is enabled (so that it will start automatically on subsequent reboots), started, and its status checked with the commands

```
[root@girtab ~]# systemctl enable snort
ln -s '/usr/lib/systemd/system/snort.service' '/etc/systemd/system/multi-user.
target.wants/snort.service'
[root@girtab system]# systemctl start snort
[root@girtab system]# systemctl status snort
snort.service - Snort Intrusion Detection System
   Loaded: loaded (/usr/lib/systemd/system/snort.service; enabled)
   Active: active (running) since Sun 2018-07-29 14:42:43 EDT; 7s ago
 Main PID: 4363 (snort)
   CGroup: /system.slice/snort.service
        └─4363 /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/etc...
```

Configuring Snort as a Service on Mint or Ubuntu

The approach to configuring Snort to start as a service on Mint or Ubuntu depends on whether the system uses Upstart or systemd to manage services.

As an example of an Upstart system, consider Ubuntu 14.04. To configure Snort to start as a service on Ubuntu 14.04, create the Upstart script `/etc/init/snort.conf` (Listing 19-3) with the following content.

Listing 19-3. Sample Upstart script `/etc/init/snort.conf` to control Snort on Ubuntu 14.04

```
description "Snort Service"
stop on runlevel [!2]
start on runlevel [2]
script
   exec /usr/local/bin/snort -A fast -b -d -D -i eth0 -u snort -g snort
-c /etc/snort/etc/snort.conf -l /var/log/snort
end script
```

This instructs Snort to run as a daemon under the user and group snort with the configuration file `/etc/snort/etc/snort.conf`. The Snort service can be started from the command line with a command like

```
cgauss@westbrook:~$ sudo service snort start
snort start/running, process 2801
```

The script is set to automatically start Snort in runlevel 2, which is the default runlevel on an Ubuntu system (*cf.* Chapter 13).

On an Ubuntu or Mint system that uses systemd like Ubuntu 17.04, the administrator proceeds by creating /lib/systemd/system/snort.service with the same content used for a CentOS 7 system using systemd (Listing 19-2).

```
cgauss@Hispania:~$ sudo systemctl enable snort
Created symlink /etc/systemd/system/multi-user.target.wants/snort.service ➤ /lib/
systemd/system/snort.service.
cgauss@Hispania:~$ sudo systemctl start snort
cgauss@Hispania:~$ sudo systemctl -l status snort
● snort.service - Snort Intrusion Detection System
   Loaded: loaded (/lib/systemd/system/snort.service; enabled; vendor preset:
   enabled)
   Active: active (running) since Sun 2018-07-29 17:31:16 EDT; 1min 42s ago
   Main PID: 4140 (snort)
    Tasks: 2 (limit: 4915)
   CGroup: /system.slice/snort.service
           └─4140 /usr/local/bin/snort -A fast -b -d -D -i enp0s3 -u snort -g
             snort -c /etc/snort/etc/snort.conf
```

Configuring Snort as a Service on OpenSuSE

On OpenSuSE systems that use systemd like OpenSuSE 13 or 42, create the file /usr/lib/systemd/system/snort.service. The content should be the same as Listing 19-2, though the interface name may need to be adjusted.

It is enabled (to start on subsequent boots), started, and its status checked with the commands

```
dschubba:~ # systemctl enable snort
Created symlink from /etc/systemd/system/multi-user.target.wants/snort.service to
/usr/lib/systemd/system/snort.service.
dschubba:~ # systemctl start snort
dschubba:~ # systemctl status snort
● snort.service - Snort Intrusion Detection System
   Loaded: loaded (/usr/lib/systemd/system/snort.service; enabled; vendor preset:
   disabled)
   Active: active (running) since Sun 2018-07-29 18:26:29 EDT; 3s ago
   Main PID: 5208 (snort)
    Tasks: 1 (limit: 512)
   CGroup: /system.slice/snort.service
           └─5208 /usr/local/bin/snort -A fast -b -d -D -i eth0 -u snort -g snort
             -c /etc/snort/etc/snort.conf -l /var/log/snort
```

On systems like OpenSuSE 12.1, the administrator can use a SysVInit script. The script that is included with the Snort source is not optimized for use on OpenSuSE systems. The Snort documentation page https://www.snort.org/documents contains startup scripts for a range of operating systems, including OpenSuSE 12.x (https://www.snort.org/documents/snort-startup-script-for-opensuse-12-x). Download and install the startup script in /etc/init.d/snortd and the configuration file in /etc/sysconfig/snort, updating the location of the snort.conf configuration file in both scripts. Once the changes are made, the service can be started.

```
vinogradov:~ # service snortd start
redirecting to systemctl
vinogradov:~ # service snortd status
redirecting to systemctl
snortd.service - LSB: Start snort
  Loaded: loaded (/etc/init.d/snortd)
  Active: active (running) since Sun, 08 Mar 2015 10:56:18 -0400; 2s ago
  Process: 2789 ExecStart=/etc/init.d/snortd start (code=exited,
           status=0/SUCCESS)
  CGroup: name=systemd:/system/snortd.service
          └ 2799 /usr/local/bin/snort -b -d -D -i eth0 -u snort -g ...
```

Snort can be set to start on boot with YaST by navigating to System Services (Runlevel).

Snort as a Windows Service

Snort can be configured to run as a service on a Windows system. From an administrator command prompt in the directory containing the Snort binary, run the command

```
c:\Snort\bin>snort /service /install -c C:\Snort\etc\snort.conf -l C:\Snort\log

 [SNORT_SERVICE] Attempting to install the Snort service.

 [SNORT_SERVICE] The full path to the Snort binary appears to be:
    c:\Snort\bin\snort /SERVICE

 [SNORT_SERVICE] Successfully added registry keys to:
    \HKEY_LOCAL_MACHINE\SOFTWARE\Snort\

 [SNORT_SERVICE] Successfully added the Snort service to the Services database.
```

The name of the Snort service is snortsvc. To configure the Snort service to start on boot, run the command

```
c:\Snort\bin>sc config snortsvc start= delayed-auto
[SC] ChangeServiceConfig SUCCESS
```

To start the Snort service, run

```
c:\Snort\bin>sc start snortsvc

SERVICE_NAME: snortsvc
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE,
                                 IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 3892
        FLAGS              :
```

The graphical tool to manage services can be launched by navigating Control Panel ➤ System and Security ➤ Administrative Tools ➤ Services (Figure 4-3). Double-clicking on the service (Snort) allows the service to be started, stopped, and/or configured to run on system start (Figure 19-1).

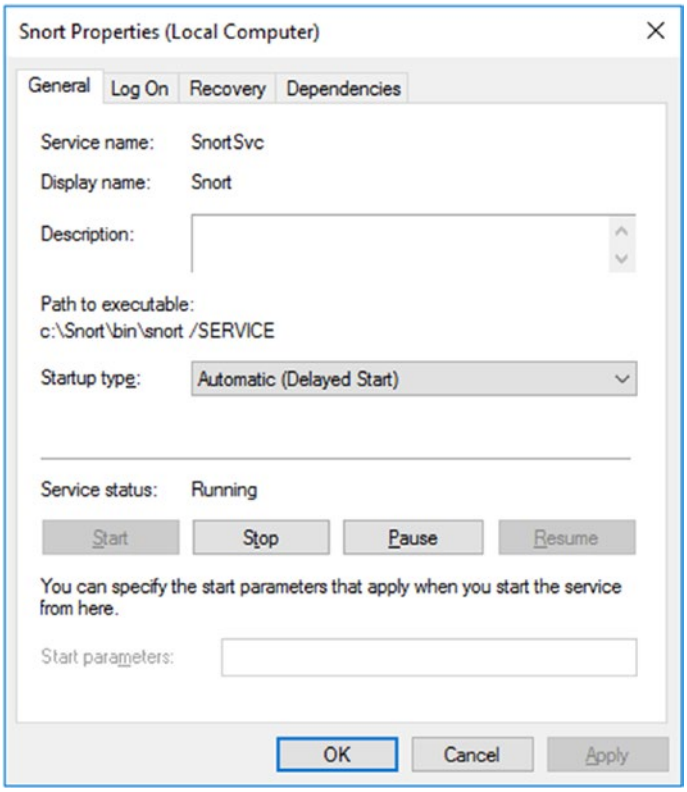

Figure 19-1. *Graphical tool to manage the Snort service. Windows 10-1607.*

Snort Variables and Preprocessors

Snort, like all IDSs, must be tuned - this reduces the number of false positives the system generates as well as ensures that traffic is being analyzed correctly. This configuration takes place in the Snort configuration file `snort.conf`.

Snort Variables

Section 1 of the configuration file `snort.conf` sets up the network variables. It starts by defining the home network, which is the address space the IDS is defending. For example, suppose that the administrator's network is in the address space 10.0.11.0/24, with a DNS server at 10.0.11.0 and web servers at 10.0.11.12 and 10.0.11.13. One reasonable starting point are the declarations

```
# Setup the network addresses you are protecting
ipvar HOME_NET 10.0.11.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS 10.0.11.10

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS [10.0.11.12,10.0.11.13]
```

Here the home network is set to the full address space 10.0.11.0/24, while the external network is set to all addresses outside the home network. The location of the single DNS server is specified, as are the locations of both web servers. This example does not contain an SMTP server; however, the variable `SMTP_SERVERS` must be set as it is used in a range of rule sets like `rules/browser-firefox.rules`.

Following the address variable declarations are port variable declarations; these can generally be left in their default state. As examples of these directives are the following, which provide Snort the ports used for SSH and FTP servers.

```
# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]
```

The first section concludes with the location of the rule files

```
var RULE_PATH ../rules
var SO_RULE_PATH ../so_rules
var PREPROC_RULE_PATH ../preproc_rules

# If you are using reputation preprocessor set these
var WHITE_LIST_PATH ../rules
var BLACK_LIST_PATH ../rules
```

These default to a relative path from the snort configuration file `snort.conf`. Provided the configuration file has not been moved from its initial location `/etc/snort/etc/snort.conf`, these variables point to the proper locations of the rules.

Snort Decoders

Section 2 of the configuration file `snort.conf` begins with rules that configure various decoders. When a packet is received by Snort, it is decoded to determine the basic properties of the packet, like its type and protocol. The decoder may spawn an alert if the packet is malformed; this process is configurable. The various options are described in the file `/usr/local/src/snort-2.9.11.1/doc/README.decode` that is included with the Snort source code.

Further in section 2 is a commented line to configure the maximum number of flowbits

```
# Configure maximum number of flowbit references.  For more information, see
README.flowbits
# config flowbits_size: 64
```

Flowbits are a way for Snort to relate the contents of one packet to another. A rule can see a pattern in a packet, and then set a flowbit. If another rule sees another pattern and if that flowbit is set, then the second rule can fire an alert. Do not simply uncomment the line, however, as current rule sets use more than 64 flowbits; the default allows 1024 flowbits.

Section 2 concludes with some additional options, including options for Snort running inline or in an active response mode; to be used these features need to be included at compile time.

Section 3 of the configuration file `snort.conf` provides technical configuration, including the algorithms used in the detection engine. These can be left in their default states.

Snort Rule Locations

Section 4 sets the paths to the preprocessor library, the preprocessor engine, and the dynamic rules libraries.

```
# path to dynamic preprocessor libraries
dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/
```

```
# path to base preprocessor engine
dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so

# path to dynamic rules libraries
dynamicdetection directory /usr/local/lib/snort_dynamicrules/
```

Recall that during rule installation, `/usr/local/lib/snort_dynamicrules/` is configured as a symbolic link that points to the subdirectory of `/etc/snort/so_rules/precompiled/` that matches the operating system, the system architecture, and the running version of Snort. On OpenSuSE systems or on Windows systems, some or all three path variables may need to be modified.

IP Fragmentation Preprocessor

Section 5 of the snort configuration file configures the preprocessors. Preprocessors run after packets are decoded, but before intrusion detection. One of the first preprocessors configured is frag3.

One approach to evading an IDS is to fragment the packets. Individually the different fragments may be inoffensive, but when reassembled they are malicious. Different operating systems may reassemble fragmented packets in different ways; this is especially the case when the fragmented packets are malformed. The frag3 preprocessor reassembles fragmented packets so that they can be evaluated. The default policy in the configuration file is to assume that the targets in the home network are Windows systems. It is possible to configure a default policy and then override it for the specific IP addresses 10.0.11.10 and 10.0.11.12 (that are running Linux) with directives like

```
# Target-based IP defragmentation.  For more inforation, see README.frag3
preprocessor frag3_global: max_frags 65536

preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10 min_
fragment_length 100 timeout 180

preprocessor frag3_engine: bind_to [10.0.11.10,10.0.11.12] policy linux detect_
anomalies overlap_limit 10 min_fragment_length 100 timeout 180
```

Stream Preprocessor

Snort passes TCP and UDP traffic through the stream5 preprocessor, and like the frag3 preprocessor, the TCP reassembly process depends on the underlying operating system. Unlike the frag3 preprocessor, though, the stream5_tcp preprocessor does not accept lists of addresses in its `bind_to` configuration option. If all the defended hosts are Windows save for Linux systems on 10.0.11.10 and 10.0.1.12, one approach is to use the following configuration.

```
# Target-Based stateful inspection/stream reassembly.  For more inforation, see
README.stream5

preprocessor stream5_global: track_tcp yes, track_udp yes, track_icmp no, max_tcp
262144, max_udp 131072, max_active_responses 2, min_response_seconds 5
```

```
preprocessor stream5_tcp: policy windows, detect_anomalies, require_3whs 180,
overlap_limit 10, small_segments 3 bytes 150, timeout 180

... Output Deleted ...

preprocessor stream5_tcp: bind_to 10.0.11.10, policy linux, detect_anomalies,
require_3whs 180, overlap_limit 10, small_segments 3 bytes 150, timeout 180

preprocessor stream5_tcp: bind_to 10.0.11.12, policy linux, detect_anomalies,
require_3whs 180, overlap_limit 10, small_segments 3 bytes 150, timeout 180

preprocessor stream5_udp: timeout 180
```

HTTP Preprocessor

Traffic to or from web servers is processed by the HTTP preprocessor. The http_inspect preprocessor can be tuned differently for different types of web servers by the profile directive. Available values include all, apache, iis, iis5_0, and iis4_0. The http_inspect preprocessor only decodes traffic on the ports specified. HTTPS traffic is encrypted and cannot be decoded with http_inspect; thus port 443 and other SSL protected ports should not be included in the list of ports for http_inspect. Given an IIS server on 10.0.11.13 and an Apache server on 10.0.11.12, a reasonable configuration would be

```
# HTTP normalization and anomaly detection.  For more information, see README.
http_inspect
preprocessor http_inspect: global iis_unicode_map unicode.map 1252 compress_depth
65535 decompress_depth 65535

preprocessor http_inspect_server: server { 10.0.11.12 } profile apache \
  ports { 80 } extended_response_inspection enable_cookie inspect_gzip \
  unlimited_decompress normalize_javascript server_flow_depth 0 \
  client_flow_depth 0 post_depth 65495 allow_proxy_use \
  oversize_dir_length 300 normalize_headers normalize_cookies \
  normalize_utf max_headers 100

preprocessor http_inspect_server: server default profile iis \
  ports { 80 } extended_response_inspection enable_cookie inspect_gzip \
  unlimited_decompress normalize_javascript server_flow_depth 0 \
  client_flow_depth 0 post_depth 65495 allow_proxy_use \
  oversize_dir_length 300 normalize_headers normalize_cookies \
  normalize_utf max_headers 100
```

Here the default profile is for IIS, which is overridden for 10.0.11.12.

Unnecessary Preprocessors

Care should be taken when selecting and including preprocessors, and unnecessary ones should not be enabled. The default Snort configuration enables a preprocessor to detect Back Orifice; this is a remote access trojan released in the late 1990s. In 2005, a vulnerability (CVE 2005-3252) was discovered in the Back Orifice preprocessor; there is a corresponding Metasploit exploit (`exploit/linux/ids/snortopre`) that affects Snort 2.4.0–2.4.3.

Port Scan Detection

One useful preprocessor is sfportscan, which is used to detect port scans like NMap. Configure it in the file `snort.conf` with a line like

```
# Portscan detection.  For more information, see README.sfportscan
preprocessor sfportscan: proto { all } memcap { 10000000 }
sense_level { medium } logfile { pscan }
```

If a port scan is detected, it is recorded in the file `/var/log/snort/pscan`. After a port scan, that file contains alerts like

```
dschubba:/var/log/snort # cat pscan
Time: 07/29-22:22:40.027310
event_ref: 0
10.0.2.2 -> 10.0.2.90 (portscan) TCP Filtered Portscan
Priority Count: 1
Connection Count: 200
IP Count: 1
Scanner IP Range: 10.0.2.2:10.0.2.2
Port/Proto Count: 199
Port/Proto Range: 21:50002
```

ARP Spoof Detection

Another useful preprocessor is arpspoof; this can be used to detect ARP spoofing attacks. Suppose that an administrator knows that the MAC address 08:00:27:16:3f:a8 is assigned to a host with the IP address 10.0.3.48 and that the MAC address 08:00:27:ea:86:0d is assigned to a host with the IP address 10.0.3.43. Then the administrator can uncomment the ARP spoof detection directives in `snort.conf` and configure the preprocessor.

```
# ARP spoof detection.  For more information, see the Snort Manual - Configuring
Snort - Preprocessors - ARP Spoof Preprocessor
preprocessor arpspoof: -unicast
preprocessor arpspoof_detect_host: 10.0.3.48 08:00:27:16:3f:a8
preprocessor arpspoof_detect_host: 10.0.3.43 08:00:27:ea:86:0d
```

The first directive instructs Snort to detect unicast ARP requests; the remaining directives match the IP addresses to their MAC addresses.

Enabling the preprocessor is necessary but not sufficient to generate alerts; the corresponding rule set (`preproc_rules/preprocessor.rules`) must also be enabled; by default, these rules are commented out. Provided the preprocessor and the rule set are enabled, then Snort detects ARP spoofing attacks with alerts of the form

```
[**] [112:1:1] (spp_arpspoof) Unicast ARP request [**]
07/30-21:59:16.185792

[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
07/30-21:59:17.406686

[**] [112:1:1] (spp_arpspoof) Unicast ARP request [**]
07/30-21:59:17.608138

[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
07/30-21:59:27.416854
```

Sensitive Data and Other Preprocessors

The SDF sensitive data preprocessor can be used to detect credit card numbers, social security numbers, email addresses, and phone numbers leaving the network. The default directive in the configuration file `snort.conf` is

```
# SDF sensitive data preprocessor.  For more information see README.sensitive_data
preprocessor sensitive_data: alert_threshold 25
```

This sets the detection threshold. Enabling the preprocessor is necessary but not sufficient to generate alerts; the corresponding rule set (`preproc_rules/sensitive-data.rules`) must also be enabled; by default, these rules are commented out.

Administrators should take the time to read and understand these rules before deployment. The sensitive data file contains six rules; as an example, the first has the content

```
alert tcp $HOME_NET any -> $EXTERNAL_NET [80,20,25,143,110] (msg:"SENSITIVE-
DATA Credit Card Numbers"; metadata:service http, service smtp, service ftp-
data, service imap, service pop3; sd_pattern:2,credit_card; classtype:sdf; sid:2;
gid:138; rev:1;)
```

This rule will alert on traffic from the home network to an external network via TCP on the ports 20, 25, 80, 110, and 143. A check of the documentation for these rules, available in the source code in the file `/usr/local/src/snort-2.9.11.1/doc/README.sensitive_data` explains that this pattern will match if two or more 15- or 16-digit credit card numbers from Visa, Mastercard, Discover, or American Express are observed. These credit card numbers have their

checksums checked for validity and are matched if there are either spaces or dashes between groups of numbers, or even there is nothing between the groups.

Notice the limitations in the rule. It does not detect or alert on encrypted traffic like SSL or HTTPS, as the unencrypted data is not available. This rule is unlikely to detect an attacker exfiltrating data from a defended network. Even if the attacker is not encrypting the traffic, if the port that receives the data is not one of 20, 25, 80, 110, or 143, then the rule will not fire.

However, provided the preprocessor and the rule set are enabled, then Snort detects sensitive data leaving the network with alerts of the form

```
07/31-20:39:40.103572  [**] [138:2:1] SENSITIVE-DATA Credit Card Numbers [**]
[Classification: Sensitive Data was Transmitted Across the Network] [Priority: 2]
{TCP} 10.0.3.48:53024 -> 10.0.2.2:80
```

Section 5 of the `snort.conf` configuration file contains other preprocessors that normalize many other kinds of traffic, including FTP/Telnet, ONC-RPC (for Linux systems), SMB/DCE-RPC (primarily for Windows systems), SMTP, SSH, DNS, SSL, SIP, IMAP, and POP.

Snort Output

Section 6 of the `snort.conf` configuration file includes several commented-out output plugins.

Controlling Snort Output from the Command Line

Snort output is first configured by flags passed on the command line when Snort is started. The `-A` flag can be used to specify the output mode; available options include `fast`, which writes a single line message, and `full` which includes a header. If the `-b` flag is used, then Snort stores binary packet captures. Both the alerts and the binary captures are stored in the log directory, which can be specified with the `-l` flag.

If either the `-A` or the `-b` flag are specified when Snort is started, then the output plugins in section 6 of the `snort.conf` file may be ignored. This is the approach that was taken in Listing 19-2 to launch Snort as a service via systemd on CentOS 7 and in Listing 19-3 to launch Snort as a service via Upstart on Mint or Ubuntu. To use the directives in `snort.conf` to control the output, these files need to be modified.

When Snort was configured to start as a service on CentOS 5 or 6, the script `/usr/local/src/snort-2.9.11.1/rpm/snortd` was copied to `/etc/init.d/snortd` and the configuration file `/usr/local/src/snort-2.9.11.1/rpm/snort.sysconfig` was copied to `/etc/sysconfig/snort`. The default settings in `/etc/sysconfig/snort` enable both fast logging and binary logging when Snort starts. This can be disabled by commenting out the lines from `/etc/sysconfig/snort`.

```
#ALERTMODE=fast

... Lines Omitted ...

#BINARY_LOG=1
```

On OpenSuSE 12.x, the Snort service script `/etc/init.d/snortd` and configuration file `/etc/sysconfig/snort` taken from https://www.snort.org/documents/snort-startup-script-for-opensuse-12-x enable binary logging; this can be disabled by commenting out the line from the copied `/etc/sysconfig/snort`.

```
#BINARY_LOG=1
```

Snort Syslog Logging

Provided the output is not configured from the command line, one available output plugin is to use Syslog. To send alerts to syslog with facility `auth` and priority `alert`, use the `snort.conf` directive

```
output alert_syslog: LOG_AUTH LOG_ALERT
```

Alerts then appear in the system logs in the general format

```
Jul 31 22:40:24 scheat snort[3199]: [1:1000001:1] Snort Shibboleth Testing Rule
{TCP} 10.0.3.48:80 -> 10.0.2.94:36394
```

Snort Unified Log

Another option is the unified output format, which is generated with `snort.conf` directives like the following:

```
output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, vlan_
event_types
```

This stores the alerts in the file `merged.log` in the log directory. Unified format is a binary format, so the result cannot simply be viewed in a text editor. However, the tool `u2spewfoo`[9] can be used to print the results in a human readable format

```
[root@scheat ~]# u2spewfoo /var/log/snort/merged.log

(Event)
        sensor id: 0    event id: 1    event second: 1533091412    event
           microsecond: 444963
    sig id: 1000001    gen id: 1       revision: 1      classification: 0
        priority: 0     ip source: 10.0.3.48     ip destination: 10.0.2.94
        src port: 80    dest port: 36396         protocol: 6    impact_flag: 0
           blocked: 0
        mpls label: 0   vland id: 0     policy id: 0
```

[9]What a sense of humor.

```
Packet
        sensor id: 0    event id: 1     event second: 1533091412
        packet second: 1533091412       packet microsecond: 444963
        linktype: 1     packet_length: 590
[ 0] 08 00 27 79 FC B6 08 00 27 16 3F A8 08 00 45 00  ..'y....'.?...E.
[ 16] 02 40 82 8A 40 00 40 06 9C A0 0A 00 03 30 0A 00  .@..@.@......0..
[ 32] 02 5E 00 50 8E 2C CF 9D 2F 26 CA D1 E9 74 80 18  .^.P.,../&...t..
[ 48] 00 E3 CF 6F 00 00 01 01 08 0A 00 38 AD AF 00 E2  ...o.......8....
[ 64] 49 F7 48 54 54 50 2F 31 2E 31 20 32 30 30 20 4F  I.HTTP/1.1 200 O
[ 80] 4B 0D 0A 44 61 74 65 3A 20 57 65 64 2C 20 30 31  K..Date: Wed, 01
[ 96] 20 41 75 67 20 32 30 31 38 20 30 32 3A 34 33 3A   Aug 2018 02:43:

... Output Deleted ...
```

Snort Rules

Sections 7, 8, and 9 of the `snort.conf` configuration file include directives that incorporate the various rules. These rules are split into separate files as an organizational aide.

Once the configuration file is tuned, it should be checked; this can be done by running Snort with the configuration file and the `-T` flag.

```
[root@scheat ~]# snort -T -c /etc/snort/etc/snort.conf
Running in Test mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/etc/snort.conf"

... Output Deleted ...

Snort successfully validated the configuration!
Snort exiting
```

The output from this test is lengthy and should be checked carefully and any errors corrected. If dynamic libraries are used, Snort reports that they are loaded correctly with lines like

```
Loading dynamic engine /usr/local/lib/snort_dynamicengine/libsf_engine.so... done
Loading all dynamic detection libs from /usr/local/lib/snort_dynamicrules...
  Loading dynamic detection library /usr/local/lib/snort_dynamicrules/
  file-image.so... done
  Loading dynamic detection library /usr/local/lib/snort_dynamicrules/malware-cnc.
  so... done
```

Depending on the enabled rule set, some warnings may be displayed. For example, if a rule sets a flowbit, but no subsequent rule checks the value, the user receives warnings like

```
WARNING: flowbits key 'file.xfdl' is set but not ever checked.
```

Snort and EternalBlue

As an example of the use of Snort, suppose that a Snort sensor has been placed between an attacker (at 10.0.2.2 in this example) and a vulnerable Windows 7 SP1 64-bit system (at 10.0.15.210 in this example). Suppose also that all the rules have been enabled, and that syslog is being used to record the Snort alerts. The attacker uses the Metasploit module `exploit/windows/smb/ms17_010_eternalblue` to successfully obtain a system shell on the target.

The logs on the Snort sensor contain a log entry of the following form.

```
Aug  1 22:32:00 scheat snort[4492]: [1:42944:2] OS-WINDOWS Microsoft Windows SMB
remote code execution attempt [Classification: Attempted Administrator Privilege
Gain] [Priority: 1] {TCP} 10.0.2.2:44207 -> 10.0.15.210:445
```

This log entry shows the date and time of the entry, along with the hostname of the sensor (scheat) as well as the name and PID of the process that generated the alert (Snort with PID 4492). The log entry continues with the SID and revision for the rule; in this case the alert was caused by the rule with SID 42944, revision 2.

The alert shows the traffic from the attacker's system at 10.0.2.2 and the target at 10.0.15.210 on TCP/445.

Although the alert provides a short description of the alert (OS-WINDOWS Microsoft Windows SMB remote code execution attempt), that is sometimes insufficient to understand the reason the rule fired. To obtain more information about the rule that generated the alert, the administrator can visit `https://www.snort.org/docs` and provide the rule SID: 42944. That page provides the following information about the rule:

> *Message*
>
> *OS-WINDOWS Microsoft Windows SMB remote code execution attempt*
>
> *Summary*
>
> *The SMBv1 server in Microsoft Windows Vista SP2; Windows Server 2008 SP2 and R2 SP1; Windows 7 SP1; Windows 8.1; Windows Server 2012 Gold and R2; Windows RT 8.1; and Windows 10 Gold, 1511, and 1607; and Windows Server 2016 allows remote attackers to execute arbitrary code via crafted packets, aka "Windows SMB Remote Code Execution Vulnerability." This vulnerability is different from those described in CVE-2017-0143, CVE-2017-0145, CVE-2017-0146, and CVE-2017-0148.*

... Output Deleted ...

Additional References
isc.sans.edu/forums/diary/ETERNALBLUE+Possible+Window+SMB+Buffer+Overflow+0Day/22304/
technet.microsoft.com/en-us/security/bulletin/MS17-010

https://www.snort.org/rule_docs/1-42944

The actual rule for the alert can be examined; it is in the file `/etc/snort/rules/os-windows.rules`. It has the content

```
alert tcp any any -> $HOME_NET 445 (msg:"OS-WINDOWS Microsoft Windows SMB remote
code execution attempt"; flow:to_server,established; content:"|FF|SMB|A0 00 00
00 00|"; depth:9; offset:4; content:"|01 00 00 00 00|"; within:5; distance:59;
byte_test:4,>,0x8150,-33,relative,little; metadata:policy balanced-ips drop,
policy connectivity-ips drop, policy max-detect-ips drop, policy security-
ips drop, ruleset community, service netbios-ssn; reference:cve,2017-0144;
reference:cve,2017-0146; reference:url,isc.sans.edu/forums/diary/ETERNALBLUE+Poss
ible+Window+SMB+Buffer+Overflow+0Day/22304/; reference:url,technet.microsoft.com/
en-us/security/bulletin/MS17-010; classtype:attempted-admin; sid:42944; rev:2;)
```

This shows the content that is used to trigger the alert, along with a collection of references, both to the CVE number of the vulnerability as well as to web pages that provide more information.

Notes and References

The best place to go for current documentation for Snort is the Snort manual, online at http://manual.snort.org/.

Snort is also included in Security Onion (https://code.google.com/p/security-onion/, http://blog.securityonion.net/), a Linux distribution designed for intrusion detection. An excellent book that covers not just Security Onion, but the entire process of monitoring a network for intrusions is

- *The Practice of Network Security Monitoring: Understanding Incident Detection and Response,* Richard Bejtlich. No Starch Press, August 2013.

That book is a worthy successor to the older book

- *The Tao of Network Security Monitoring: Beyond Intrusion Detection,* Richard Bejtlich. Addison-Wesley, July 2004.

The differences between the official Snort rule sets is explained at http://blog.snort.org/2014/07/snort-subscriber-rule-set-update.html; see also https://www.snort.org/documents/57.

This chapter drops the rules in the subdirectories of `/etc/snort`, with the configuration file in `/etc/snort/etc/snort.conf`. One of the rationales in favor of this choice is that it simplifies the exposition, as the rules are in the same place regardless of the distribution; it also means that fewer changes need to be made to the provided `snort.conf` file before Snort can be first run. On the other hand, it would also make sense for Snort configuration to be `/etc/snort.conf`, and the rules stored in `/usr/local/lib`.

The installation of Snort on Mint 17.3 can be somewhat problematic because of the way it manages packages. For example, the system may be unable to install `libpcre3-dev` (in part) because version 1:8.31-2ubuntu2 for `libpcre` is required while version 1:8.31-2ubuntu2.1 is to be installed. This can be resolved with aptitude in the same way BIND was installed on Mint 18.1; see the Notes and References for Chapter 4.