Snort 2.9.8.x on Ubuntu 12, 14, and 15

with Barnyard2, PulledPork, and Snorby

Noah Dietrich
Noah@SublimeRobots.com

December 16, 2015

Contents

1 Introduction

This guide will walk you through installing Snort as a NIDS (network intrusion detection system), with three pieces of additional software to improve the functionality of Snort. This guide is written with the Snort host as a VMware vSphere virtual machine, but can be easily used to install Snort on a physical machine or as a virtual machine on another platform.

The latest version of this guide plus additional notes can be found at SublimeRobots.com.

This installer guide has been tested on the following versions of Ubuntu running on VMware vSphere 3:

- Ubuntu 12.04.5 LTS x86
- Ubuntu 12.04.5 LTS x64
- Ubuntu 14.04.03 Server LTS x86
- Ubuntu 14.04.03 Server LTS x64
- Ubuntu 15.10 Server x86
- Ubuntu 15.10 Server x64

While you can choose to install Snort without any supporting software and it will work just fine, it becomes much more useful with a few additional software packages. These packages are:

Barnyard2:
 Software that takes Snort output and writes to a SQL database, which reduces load on the system.

PulledPork:
 Automatically downloads the latest Snort rules.

Snorby:
 A web-based graphical interface for viewing and clearing Snort events.

If you just want to setup Snort on a Ubuntu system without going through the work in this document, there is a project called Autosnort that will install all the same software as this guide with a script. Optionally, you could use a fully configured LiveCD like EasyIDS or Security Onion. The benefit of this guide over *Autosnort*, *EasyIDS*, or *Security Onion* is that this guide walks you through installing each component, explaining the steps as you go along. This will give you a better understanding of the software components that make up Snort, and will allow you to configure Snort for your own needs.

Note: while this guide focuses on the current 2.9.8.x series release of Snort, these steps will most likely work to install the older Snort 2.9.7.x series, and could be used to install Snort on older or derivative versions of Ubuntu (Xubuntu, Mint, etc.). I have also been told that these instructions are helpful for installing Snort on Debian systems, but I haven't verified that myself.

2 About This Guide

Passwords: This guide chooses to use simplistic passwords to make it obvious as to what is being done. You should select your own secure passwords in place of these passwords.

Software Package Versions: This guide is written to install with the latest version of all software available, except where noted for compatibility reasons. This guide should work with slightly newer or older versions of all software packages, but ensuring compatibility is up to the individual user. If you have issues when installing a different version of any software than what this guide uses, I recommend that you try installing the exact version this guide uses in order to determine if the error is with the specific software version or

is due to a different issue. Additionally, this guide tries to use software from official Ubuntu repositories as much as possible, only downloading software from trusted 3rd party sites (such as snort.org only when no package is available from official repositories.

Software versions used in this guide:

- Snort 2.9.8.0
- Barnyard2 2-1.14
- PulledPork 0.7.2
- Snorby 2.6.2

Administrator Accounts: This guide assumes that you are logged into the system as a normal user, and will run all administrative commands with `sudo`. This helps to identify what commands require administrative credentials, and which do not. We will also create a non-privileged user named `snort` that will be used to run all applications when setting up services, following current best security practices.

3   Enabling OpenAppID

If you are interested in adding OpenAppID support to Snort, please see this article on my blog. For more information about OpenAppID, please see Firing up OpenAppID.

4   Enviornment

As stated above, this guide was written geared towards installing Snort as a virtual machine running on an VMware vSphere 3 hypervisor. The vSphere hypervisor is a free product from vMware, and which I highly recommend for testing software due to the ability to create snapshots. If you choose to install Snort outside of a virtual machine, the steps below should be the same, except for a few VMware specific steps that should be fairly obvious once you've worked through this guide.

5   Ethernet Interface Names On Ubuntu 15.10

Important note for people running Ubuntu 15.10: In Ubuntu 15.10, for new installations (not upgrades), network interfaces no longer follow the `ethX` standard (eth0, eth1, ...). Instead, interfaces names are assigned as Predictable Network Interface Names. This means you need to check the names of your interfaces using `ifconfig`, since you will need to reference the name of your interface for many steps in this guide. In my case, what was originally **eth0** is now **ens160**. If you are running Ubuntu 15.10, anywhere in this guide you see eth0, you will need to replace with your new interface name.

6   VMware Virtual Machine Configuration

If you are using VMware vSphere to host your Snort virtual machine, when creating the virtual machine, make sure to select the **VMXNET 3** network adapter (not the default adapter) when creating the client virtual machine, as it works better for Snort[1] [2].

---

[1]https://isc.sans.edu/diary/Running+Snort+on+VMWare+ESXi/15899
[2]http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1001805

This guide assumes that you have created a virtual machine with a single network adapter that will be used for both administrative control (over SSH) as well as for Snort to listen on for traffic. You can easily add more adapters when setting up the system or at a later date, you just need to make sure to specify the correct adapter Snort should listen on at runtime (this should be fairly obvious).

7   Installing Ubuntu

This guide will assume that you have installed one of the supported versions of Ubuntu with all the default settings, and that you have selected "install security updates automatically" during the configuration.

Snort does not need an ip address assigned to the interface that it is listening on, however it makes it easier to manage the system remotely via ssh if an interface is reachable. In a production environment, it is recommended that you user one interface on your Snort server for management, and have Snort listen on other interfaces, but this is not required. By default Ubuntu will use DHCP to auto-configure an address, if this is the case, you can verify your ip address by running `ifconfig eth0`. If you do not have a DHCP server assigning IP addresses, configure one on your Snort system manually. You will need internet connectivity in order to download the required packages and software tarballs.

Once you have logged in for the first time and verified internet connectivity, make sure the system is up to date, and install openssh-server (so we can remotely-manage the system). Reboot after installation to make sure all patches are applied.

```
# Install Updates and reboot:
sudo apt-get update
sudo apt-get dist-upgrade -y
sudo apt-get install -y openssh-server
sudo reboot
```

If you are installing Snort on a VMware vSphere server, I recommend installing the VMware tools as well. Instructions can be found on VMware's Website, under the section titled: *Ubuntu Server with only a command line interface.*

8   Network Card Configuration

From http://manual.snort.org/node7.html:

> Some network cards have features named "Large Receive Offload" (lro) and "Generic Receive Offload" (gro). With these features enabled, the network card performs packet reassembly before they're processed by the kernel. By default, Snort will truncate packets larger than the default snaplen of 1518 bytes. In addition, LRO and GRO may cause issues with Stream5 target-based reassembly. We recommend that you turn off LRO and GRO.

To disable LRO and GRO for any interface that Snort listens on, we will use the `ethtool` command in the network interface configuration file `/etc/network/interfaces`. If you are running Ubuntu 12, you will need to first install ethtool:

```
sudo apt-get install -y ethtool
```

Use vi to edit the network interfaces file:

```
sudo vi /etc/network/interfaces
```

Append the following two lines for each network interface, making sure to change eth0 to match the interface you are working on, since your interface names may be different, especially on Ubuntu 15.10:

```
post-up ethtool -K eth0 gro off
post-up ethtool -K eth0 lro off
```

an example of how the `/etc/network/interfaces` file should look for a single interface:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp
post-up ethtool -K eth0 gro off
post-up ethtool -K eth0 lro off
```

Restart networking (replace eth0 with your interfaces with below) and verify that LRO and GRO are disabled:

```
user@snortserver:~$ sudo ifconfig eth0 down && sudo ifconfig eth0 up
user@snortserver:~$ ethtool -k eth0 | grep receive-offload
generic-receive-offload: off
large-receive-offload: off
user@snortserver:~$
```

if the interfaces do not show LRO and GRO as off, reboot and check again (it can be difficult to get Ubuntu to reload the network configuration without a reboot).

9 Installing the Snort Pre-Requisites

Snort has four main pre-requisites:

pcap	(libpcap-dev)	available from the Ubuntu repository
PCRE	(libpcre3-dev)	available from the Ubuntu repository
Libdnet	(libdumbnet-dev)	available from the Ubuntu repository
DAQ	(http://www.snort.org/downloads/)	compiled from source

First we want to install all the tools required for building software. The `build-essentials` package does this for us:

```
sudo apt-get install -y build-essential
```

Once our build tools are installed, we install all Snort pre-requisites that are available from the Ubuntu repositories[3]:

```
sudo apt-get install -y libpcap-dev libpcre3-dev libdumbnet-dev
```

---

[3]Many guides that install Snort on Ubuntu have you download libdnet from its homepage http://libdnet.sourceforge.net/. This is possible and will work fine. However, the `libdumbnet-dev` Ubuntu package provides the same software (do *not* install the libdnet package from Ubuntu archives, as it is an un-related package and does not provide the required libdent libraries). If you want to compile the libdent libraries from source and you are running a 64-bit version Ubuntu, use the `-fPIC` flag during the 'configure' stage.

In this guide, we will be downloading a number of tarbals for various software packages. We will create a folder called `snort_src` to keep them all in one place:

```
mkdir ~/snort_src
cd ~/snort_src
```

The Snort DAQ (Data AcQuisition library)has a few pre-requisites that need to be installed:

```
sudo apt-get install -y bison flex
```

Download and install the latest version of DAQ from the Snort website. The steps below use wget to download version 2.0.6 of DAQ, which is the latest version at the time of writing this guide.

```
cd ~/snort_src
wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz
tar -xvzf daq-2.0.6.tar.gz
cd daq-2.0.6
./configure
make
sudo make install
```

10   Installing Snort

To install Snort on Ubuntu, there is one additional required pre-requisite that needs to be installed that is not mentioned in the documentation: `zlibg` which is a compression library.

There are three optional libraries that improves fuctionality: `liblzma-dev` which provides decompression of swf files (adobe flash), `openssl`, and `libssl-dev` which both provide SHA and MD5 file signatures:

```
sudo apt-get install -y zlib1g-dev liblzma-dev openssl libssl-dev
```

We are now ready to download the Snort source tarball, compile, and then install. The `--enable-sourcefire` option gives Packet Performance Monitoring (PPM)[4] [5], which lets us do performance monitoring for rules and pre-processors, and builds Snort the same way that the Snort team does:

```
cd ~/snort_src
wget https://snort.org/downloads/snort/snort-2.9.8.0.tar.gz
tar -xvzf snort-2.9.8.0.tar.gz
cd snort-2.9.8.0
./configure --enable-sourcefire
make
sudo make install
```

If you are interested in seeing the other compile-time options that are available, run `./configure --help` to get a list of all compile-time options. The Snort team has tried to ensure that the default settings are good for most basic installations, so you shouldn't need to change anything unless you are trying to do something special.

Run the following command to update shared libraries (you'll get an error when you try to run Snort if you skip this step):

```
sudo ldconfig
```

---

[4]`--enable-sourcefire`: http://blog.snort.org/2011/09/snort-291-installation-guide-for-centos.html
[5]PPM: http://manual.snort.org/node221.html

Place a symlink to the Snort binary in /usr/sbin:

```
sudo ln -s /usr/local/bin/snort /usr/sbin/snort
```

Test Snort by running the binary as a regular user, passing it the -V flag (which tells Snort to verify itself and any configuration files passed to it). You should see output similar to what is shown below (although exact version numbers may be slightly different):

```
user@snortserver:~$ snort -V

   ,,_        -*> Snort! <*-
  o"  )~    Version 2.9.8.0 GRE (Build 229)
   ''''     By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
            Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
            Copyright (C) 1998-2013 Sourcefire, Inc., et al.
            Using libpcap version 1.5.3
            Using PCRE version: 8.31 2012-07-06
            Using ZLIB version: 1.2.8

user@snortserver:~$
```

11   Configuring Snort to Run in NIDS Mode

Since we don't want Snort to run as root, we need to create an unprivileged account and group for the daemon to run under (snort:snort). We will also create a number of files and directories required by Snort, and set permissions on those files. Snort will have the following directories: Configurations and rule files in /etc/snort Alerts will be written to /var/log/snort Compiled rules (.so rules) will be stored in /usr/local/lib/snort_dynamicrules

```
# Create the snort user and group:
sudo groupadd snort
sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort

# Create the Snort directories:
sudo mkdir /etc/snort
sudo mkdir /etc/snort/rules
sudo mkdir /etc/snort/rules/iplists
sudo mkdir /etc/snort/preproc_rules
sudo mkdir /usr/local/lib/snort_dynamicrules
sudo mkdir /etc/snort/so_rules

# Create some files that stores rules and ip lists
sudo touch /etc/snort/rules/iplists/black_list.rules
sudo touch /etc/snort/rules/iplists/white_list.rules
sudo touch /etc/snort/rules/local.rules
sudo touch /etc/snort/sid-msg.map

# Create our logging directories:
sudo mkdir /var/log/snort
sudo mkdir /var/log/snort/archived_logs

# Adjust permissions:
sudo chmod -R 5775 /etc/snort
sudo chmod -R 5775 /var/log/snort
sudo chmod -R 5775 /var/log/snort/archived_logs
sudo chmod -R 5775 /etc/snort/so_rules
sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules
```

We want to change ownership of the files we created above as well to make sure Snort can access the files it uses:

```
# Change Ownership on folders:
sudo chown -R snort:snort /etc/snort
sudo chown -R snort:snort /var/log/snort
sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules
```

Snort needs some configuration files and the dynamic preprocessors copied from the Snort source tarball into the `/etc/snort` folder.

The configuration files are:

- classification.config
- file_magic.conf
- reference.config
- snort.conf
- threshold.conf
- attribute_table.dtd
- gen-msg.map
- unicode.map

To copy the configuration files and the dynamic preprocessors, run the following commands:

```
cd ~/snort_src/snort-2.9.8.0/etc/
sudo cp *.conf* /etc/snort
sudo cp *.map /etc/snort
sudo cp *.dtd /etc/snort

cd ~/snort_src/snort-2.9.8.0/src/dynamic-preprocessors/build/usr/local/lib/snort_dynamicpreprocessor/
sudo cp * /usr/local/lib/snort_dynamicpreprocessor/
```

We now have the following directory layout and file locations:

Snort binary file:	/usr/local/bin/snort
Snort configuration file:	/etc/snort/snort.conf
Snort log data directory:	/var/log/snort
Snort rules directories:	/etc/snort/rules
	/etc/snort/so_rules
	/etc/snort/preproc_rules
	/usr/local/lib/snort_dynamicrules
Snort IP list directories:	/etc/snort/rules/iplists
Snort dynamic preprocessors:	/usr/local/lib/snort_dynamicpreprocessor/

Our Snort directory listing looks like this:

```
user@snortserver:~$ tree /etc/snort
/etc/snort
|-- attribute_table.dtd
|-- classification.config
|-- file_magic.conf
|-- gen-msg.map
|-- preproc_rules
|-- reference.config
|-- rules
|   |-- iplists
|   |   |-- black_list.rules
|   |   |-- white_list.rules
|   |-- local.rules
|-- snort.conf
|-- so_rules
|-- threshold.conf
|-- unicode.map
```

We now need to edit Snort's main configuration file, `/etc/snort/snort.conf`. When we run Snort with this file as an argument, it tells Snort to run in NIDS mode.

We need to comment out all of the individual rule files that are referenced in the Snort configuration file, since instead of downloading each file individually, we will use PulledPork to manage our rulesets, which combines all the rules into a single file. The following line will comment out all rulesets in our `snort.conf` file:

```
sudo sed -i "s/include \$RULE\_PATH/#include \$RULE\_PATH/" /etc/snort/snort.conf
```

We will now manually change some settings in the `snort.conf` file, using your favourite editor:

```
sudo vi /etc/snort/snort.conf
```

Change the following lines to meet your environment:

Line 45, `HOME_NET` should match your internal (friendly) network. In the below example our HOME_NET is 10.0.0.0 with a 24-bit subnet mask (255.255.255.0)[6]:

```
ipvar HOME_NET 10.0.0.0/24
```

Note: You should not set `EXTERNAL_NET` to `!$HOME_NET` as recommended in some guides, since it can cause Snort to miss alerts.

Note: it is vital that your HOME_NET match the IP subnet of the interface that you want Snort to listen on. Please use `ifconfig eth0 | grep "inet add"` to ensure you have the right address and mask set. Often this will be a 192.168.1.x or 10.0.0.x IP address.

Set the following file paths in snort.conf, beginning at line 104:

```
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

var WHITE_LIST_PATH /etc/snort/rules/iplists
var BLACK_LIST_PATH /etc/snort/rules/iplists
```

---

[6] http://books.gigatux.nl/mirror/snortids/0596006616/snortids-CHP-5-SECT-1.html

In order to make testing Snort easy, we want to enable the `local.rules` file, where we can add rules that Snort can alert on. Un-comment (remove the hash symbol) from line 545 so it looks like this:

```
include $RULE_PATH/local.rules
```

Once the configuration file is ready, we will have Snort verify that it is a valid file, and all necessary files it references are correct. We use the `-T` flag to test the configuration file, the `-c` flag to tell Snort which configuration file to use, and `-i` to specify the interface that Snort will listen on (this is a new requirement for the 2.9.8.x version of snort). Run `sudo snort -T -c /etc/snort/snort.conf -i eth0`. Run this command as shown below and look for the following output (only the last few lines of the output are shown for clarity):

```
user@snortserver:~$ sudo snort -T -i eth0 -c /etc/snort/snort.conf
    (...)
    Snort successfully validated the configuration!
    Snort exiting
user@snortserver:~$
```

Note for Ubuntu 15.10: Interface names have changed, and are system specific (no longer listed as ethN). In the above command, you need to replace `eth0` with the name of your interface, as shown with the `ifconfig` command (in my case it is `ens160`).

It is a good idea to scroll up through the output from this command to get a feel for what Snort is loading. A lot of it won't make sense at this time, but it will become more clear as you work more with Snort. Look for any errors and warnings listed.

12 Writing a Simple Rule to Test Snort Detection

At this stage, Snort does not have any rules loaded (our rule files referenced in `snort.conf` are empty). You can verify that Snort has not loaded any rules if you scroll up through the output from the previous command and look for: `0 Snort rules read`. To test Snort's detection abilities, let's create a simple rule that will cause Snort to generate an alert whenever Snort sees an ICMP "Echo request" or "Echo reply" message, which is easy to generate with the ubiquitous `ping` utility (this makes for easy testing of the rule).

Paste the following single line into the empty local rules file: `/etc/snort/rules/local.rules`:

```
alert icmp any any -> $HOME_NET any (msg:"ICMP test detected"; GID:1; sid:10000001; rev:001; classtype:icmp-
    event;)
```

Barnyard2 doesn't read meta-information about alerts from the local.rules file. Without this information, Barnyard2 won't know any details about the rule that triggered the alert, and will generate non-fatal errors when adding new rules with PulledPork (done in a later step). To make sure that barnyard2 knows that the rule we created with unique identifier 10000001 has the message "ICMP Test Detected", as well as some other information (please see this blog post for more information). We add the following line to the `/etc/snort/sid-msg.map` file:

```
1 || 10000001 || 001 || icmp-event || 0 || ICMP Test detected || url,tools.ietf.org/html/rfc792
```

When you un-commented line 545 above (`include $RULE_PATH/local.rules`) you were telling Snort that the `local.rules` file should be loaded by Snort. When Snort loads that file on start-up, it will see the rule you created, and use that rule on all traffic the interface sees. In this case, when we created the rule, we told Snort that it should generate an alert when it sees an ICMP ping.

Since we made changes to the Snort configuration, we should test the configuration file again:

```
sudo snort -T -c /etc/snort/snort.conf -i eth0
```

This time if you scroll up through the output, you will find that one rule (the one we created in `local.rules`, and loaded by the `include` directive in `snort.conf`) has been loaded:

```
        (...)
        ++++++++++++++++++++++++++++++++++++++++++++++++++
        Initializing rule chains...
        1 Snort rules read
            1 detection rules
            0 decoder rules
            0 preprocessor rules
        1 Option Chains linked into 1 Chain Headers
        0 Dynamic rules
        ++++++++++++++++++++++++++++++++++++++++++++++++++

        +-------------------[Rule Port Counts]---------------------------------
        |            tcp     udp    icmp     ip
        |    src      0       0       0       0
        |    dst      0       0       0       0
        |    any      0       0       1       0
        |     nc      0       0       1       0
        |    s+d      0       0       0       0
        +----------------------------------------------------------------------
```

Now that we know that Snort correctly loads our rule and our configuration, we can start snort in NIDS mode, and tell it to output any alerts right to the console. We will run Snort from the command line, using the following flags:

-A console	The 'console' option prints fast mode alerts to stdout
-q	Quiet mode. Don't show banner and status report.
-u snort	Run Snort as the following user after startup
-g snort	Run Snort as the following group after startup
-c /etc/snort/snort.conf	The path to our `snort.conf` file
-i eth0	The interface to listen on (change to your interface if different)

Note: If you are running Ubuntu 15.10, remember that your interface name is not eth0.

```
$ sudo /usr/local/bin/snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```

When you run this line, you will not initially see any output, however Snort is running, processing all packets that arrive on eth0 (or whichever interface you specified with the `-i` flag), comparing them to the rules it has loaded (in this case our single ICMP Ping rule), and will then print all alerts generated when a packet matches our rule to the console.

From another computer, ping the IP address of eth0 on the Snort computer (or alternately ping from the Snort host to another machine, or to its own eth0, but not loopback interface), and you should see console output similar to what is displayed below (in the below example, the Snort server is listening on eth0 with and IP address of 10.0.0.105, and the computer generating the ping is 10.0.0.59).

```
12/06-12:14:28.908206  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 10.0.0.59 -> 10.0.0.105
12/06-12:14:28.908241  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 10.0.0.105 -> 10.0.0.59
12/06-12:14:29.905893  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 10.0.0.59 -> 10.0.0.105
^C*** Caught Int-Signal
```

Use `ctrl-c` to stop Snort from running. Note that Snort has saved a copy of this information in `/var/log/snort`, with the name `snort.log.nnnnnnnnn` (the numbers may be different). At this point Snort is running correctly in NIDS mode and generating alerts.

13 Installing Barnyard2

It is resource intensive for Snort to write events in human-readable mode, either to the console or to text files, as done above. Ideally, we would like Snort events to be stored in a MySQL database so we can view, search, and profile the events. To efficiently get Snort events into a MySQL database, we use Barnyard2. Wewill configure Snort to output events in binary form to a folder, and then have Barnyard2 read those events asynchronously and insert them to our MySQL database.

First install the Barnyard2 pre-requisites:

```
sudo apt-get install -y mysql-server libmysqlclient-dev mysql-client autoconf libtool
```

The install will prompt you to create a root mysql user password. For the examples below, we will use MySqlROOTpassword. You should choose something different and more secure, and store it safely. We will also be creating a snort MySQL user account, and the password for that account will be MySqlSNORTpassword, please note the difference between these two passwords.

We need to tell snort that it should output it's alerts in a binary format (to a file) that Barnyard2 can process. To do that, edit the **/etc/snort/snort.conf** file, and after line 521 (the commented line starting with the hash sign) add the following line:

```
output unified2: filename snort.u2, limit 128
```

So that lines 520 and 521 now looks like:

```
# output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, vlan_event_types}
output unified2: filename snort.u2, limit 128
```

Note on Barnyard2 Version: In the commands below, we will be downloading a specific snapshot of Barnyard2 from github: Barnyard2 version 2.1.14 with commits from Oct 21, 2015 (this is the latest version at this time). I chose not to use the latest stable release: 2.1.13 because some patches have been added after that release that are important, and I chose not to use the Head release, because that will change after the release of this guide, and I won't have had the ability to test it. If you want, you can (and probably will want) to use the current head release of Barnyard2, but if you have issues, you can always come back and use the version I've used below which I have verified will work with the other pieces of software in this guide.

Now download and install Barnyard2 2.1.14 release 336:

```
cd ~/snort_src
wget https://github.com/firnsy/barnyard2/archive/7254c24702392288fe6be948f88afb74040f6dc9.tar.gz \
        -O barnyard2-2-1.14-336.tar.gz
tar zxvf barnyard2-2-1.14-336.tar.gz
mv barnyard2-7254c24702392288fe6be948f88afb74040f6dc9 barnyard2-2-1.14-336
cd barnyard2-2-1.14-336
autoreconf -fvi -I ./m4
```

Barnyard2 needs access to the dnet.h library, which we installed with the Ubuntu libdumbnet package earlier. However, Barnyard2 expects a different file name for this library. Create a soft link from dnet.h to dubmnet.h so there are no issues:

```
sudo ln -s /usr/include/dumbnet.h /usr/include/dnet.h
sudo ldconfig
```

Depending on your OS version (x86 or x86_64), you need to point the install to the correct MySQL library. Run *one* of the following two lines to configure the build process, depending on your architecture (if you are unsure which architecture you are running, use the `uname -m` command:

```
# Choose ONE of these two commands to run
./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu
./configure --with-mysql --with-mysql-libraries=/usr/lib/i386-linux-gnu
```

Now complete the build and install Barnyard2 to `/usr/local/bin/barnyard2`:

```
make
sudo make install
```

NOTE: If you get dnet.h errors at the `make` stage, you may need to tell the system where the dnet.h files are. Run the following commands before running `make` again (this has been occasionally reported as an issue):

Once Barnyard2 is installed, the next step is to copy and create some files that Barnyard2 requires to run:

```
cd ~/snort_src/barnyard2-2-1.14-336
sudo cp etc/barnyard2.conf /etc/snort
# the /var/log/barnyard2 folder is never used or referenced
# but barnyard2 will error without it existing
sudo mkdir /var/log/barnyard2
sudo chown snort.snort /var/log/barnyard2

sudo touch /var/log/snort/barnyard2.waldo
sudo chown snort.snort /var/log/snort/barnyard2.waldo
```

Since Barnyard2 saves alerts to our MySQL database, we need to create that database, as well as a 'snort' MySQL user to access that database. Run the following commands to create the database and MySQL user. When prompted for a password, use the `MySqlROOTpassword`. You will also be setting the MySQL snort user password in the fourth mysql command (to `MySqlSNORTpassword`), so change it there as well.

```
$ mysql -u root -p
mysql> create database snort;
mysql> use snort;
mysql> source ~/snort_src/barnyard2-2-1.14-336/schemas/create_mysql
mysql> CREATE USER 'snort'@'localhost' IDENTIFIED BY 'MySqlSNORTpassword';
mysql> grant create, insert, select, delete, update on snort.* to 'snort'@'localhost';
mysql> exit
```

We need to tell Barnyard2 how to connect to the MySQL database. Edit `/etc/snort/barnyard2.conf`, and at the end of the file add this line (changing password to the one you created above):

```
output database: log, mysql, user=snort password=MySqlSNORTpassword dbname=snort host=localhost
```

Since the password is stored in cleartext in the `barnyard2.conf` file, we should prevent other users from reading it:

```
sudo chmod o-r /etc/snort/barnyard2.conf
```

Now we want to test that Snort is writing events to the correct binary log file, and that Barnyard2 is reading those logs and writing the events to our MySQL database. We could just start both programs up in daemon mode and generate some events by pinging the interface (triggering the rule we created earlier), but it's better to test one portion at a time.

Run Snort in alert mode (the command we run below is how Snort will normally be run when we set it up as a daemon, except we aren't using the `-D` flag which causes it to run as a daemon).

```
sudo /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
```

Ping the interface eth0 from another computer, you won't see any output on the screen because Snort wasn't started with the `-A console` flag like before. Once the ping stops, type `ctrl-c` to stop Snort. you should see a new file in the `/var/log/snort` directory with following name: `snort.u2.nnnnnnnnn` (the numbers will be different because they are based on the current time. The `snort.log.nnnnnnnnn` is the output file we created when we first tested Snort. You can delete that file if you want:

```
user@snortserver:/var/log/snort$  ls -l /var/log/snort/
total 12
drwsrwxr-t 2 snort snort 4096 Nov  7 14:48 archived_logs
-rw-r--r-- 1 snort snort    0 Nov  7 19:53 barnyard2.waldo
-rw------- 1 snort snort  708 Nov  7 14:53 snort.log.1446904397
-rw------- 1 snort snort 1552 Nov  7 19:56 snort.u2.1446922585
```

We now run Barnyard2 and tell it to process the events in `snort.u2.nnnnnnnnn` and load them into the Snort database. We use the following flags with Barnyard2:

-c /etc/snort/barnyard2.conf	The path to the `barnyard2.conf` file
-d /var/log/snort	The folder to look for Snort output files
-f snort.u2	The Filename to look for in the above directory (snort.u2.nnnnnnnnn)
-w /var/log/snort/barnyard2.waldo	The location of the waldo file (bookmark file)
-u snort	Run Barnyard2 as the following user after startup
-g snort	Run Barnyard2 as the following group after startup

Run Barnyard2 with the following command:

```
sudo barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo \
-g snort -u snort
```

Note the slash at the end of the first line. This entire command is one line, but is broken into two lines because of word-wrap issues in this PDF. You can either copy and paste both lines together and they will run, or if you type the command manually, remove the newline and the trailing slash on the line. For more information on line continuation in bash, see the sub-section: *Escapes and line continuation* in Learn Linux, 101: The Linux command line from IBM Developerworks.

Barnyard2 will start up (be patient, it can take some time), and then it will process the alerts in the `/var/log/snort/snort.u2.nnnnnnnnn` file, write them to both the screen and the database, and then wait for more events to appear in the `/var/log/snort directory`. use `Ctrl-c` to stop the process. When Barnyard2 is processing the events, you should see output similar to:

```
(...)
Opened spool file '/var/log/snort/snort.u2.1389532785'
Closing spool file '/var/log/snort/snort.u2.1389532785'. Read 8 records
Opened spool file '/var/log/snort/snort.u2.1389535513'
12/06−12:14:28.908206  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 10.0.0.59 −> 10.0.0.105
12/06−12:14:28.908241  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 10.0.0.105 −> 10.0.0.59
12/06−12:14:29.905893  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 10.0.0.59 −> 10.0.0.105
12/06−12:14:29.905927  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 10.0.0.105 −> 10.0.0.59
Waiting for new data

^C*** Caught Int−Signal
```

once you press `Ctrl-c` to stop barnyard2, it will print information about the records it processed.

We now want to check the MySQL database to see if Barnyard2 wrote the events. Run the following command to query the MySQL database, you will be prompted for the MySQL Snort user password: `MySqlSNORTpassword` (not the MySQL root password):

```
mysql -u snort -p -D snort -e "select count(*) from event"
```

If successful, you will then get the following output, showing the 8 events written to the database from the ICMP request and reply packets (when you ping from a windows system, it will by default send 4 ICMP messages. If you pinged from another system the count could be different):

```
+----------+
| count(*) |
+----------+
|        8 |
+----------+
```

Congratulations, if you have similar output (count greater than 0) as above, then Snort and Barnyard2 are properly installed and configured. We will create startup scripts later to launch both applications as daemons automatically on boot up.

14   Installing PulledPork

PulledPork is a perl script that will download, combine, and install/update snort rulesets from various locations for use by Snort. If you would rather install rulesets manually, see Apendix: Installing Snort Rules Manually.

Install the PulledPork pre-requisites:

```
sudo apt-get install -y libcrypt-ssleay-perl liblwp-useragent-determined-perl
```

Note on PulledPork Version: The command below installs the 0.7.2 version of PulledPork as it was on November 12, 2015 (fixing issue #194). There are issues with the base 0.7.2 version of PulledPork that are fixed with later patches, but a version release (0.7.3) hasn't been created that includes those patches yet. I don't want to use the 0.7.2 version of PulledPork because of the issues, and I don't want to install the current Master version of PulledPork because it may change after the release of this guide, so I've compromised by linking to a current (as of the time of this writing) version that works well and won't change. As newer releases come out they should work, but you will need to test if you choose something different. If you have issues running PulledPork, you may need to install newer versions than what I'm using, as they are actively working on the code at this time.

Download and install the PulledPork perl script and configuration files:

```
cd ~/snort_src
wget https://github.com/finchy/pulledpork/archive/8b9441aeeb7e1477e5be415f27dbc4eb25dd9d59.tar.gz \
        -O pulledpork-0.7.2-196.tar.gz
tar xvfvz pulledpork-0.7.2-196.tar.gz
mv pulledpork-8b9441aeeb7e1477e5be415f27dbc4eb25dd9d59 pulledpork-0.7.2-196
cd pulledpork-0.7.2-196/
sudo cp pulledpork.pl /usr/local/bin
sudo chmod +x /usr/local/bin/pulledpork.pl
sudo cp etc/*.conf /etc/snort
```

Check that PulledPork runs by checking the version, using the -V flag:

```
user@snortserver:~$ /usr/local/bin/pulledpork.pl -V
PulledPork v0.7.2 - E.Coli in your water bottle!

user@snortserver:~$
```

15   Configuring PulledPork to Download Rulesets

There are a few rulesets (groups of rules for Snort) that PulledPork can download. You can configure PulledPork to download the free blacklist from Talos and the free community ruleset from Snort without creating a free snort.org account. However, if you want to download the regular rules and documentation for those rules, you need to create a free account on http://snort.org in order to get a unique Oinkcode that will allow you to download these newer rulesets.

I recommend you create a snort.org account and get an oinkcode before continuing. Keep this oinkcode private.

Configure PulledPork by editing `/etc/snort/pulledpork.conf` with the following command:

```
sudo vi /etc/snort/pulledpork.conf
```

Anywhere you see <oinkcode>enter the oinkcode you received from snort.org (if you didn't get an oinkcode, you'll need to comment out lines 19 and 26):

```
Line 19 & 26: enter your oinkcode where appropriate (or comment out if no oinkcode)

Line 29: Un-comment for Emerging threats ruleset (not tested with this guide)

Line 74: change to: rule_path=/etc/snort/rules/snort.rules
Line 89: change to: local_rules=/etc/snort/rules/local.rules
Line 92: change to: sid_msg=/etc/snort/sid-msg.map
Line 96: change to: sid_msg_version=2

Line 119: change to: config_path=/etc/snort/snort.conf

Line 133: change to: distro=Ubuntu-12-04

Line 141: change to: black_list=/etc/snort/rules/iplists/black_list.rules
Line 150: change to: IPRVersion=/etc/snort/rules/iplists
```

We want to run PulledPork manually this one time to make sure it works. The following flags are used with PulledPork:

| -l | Write detailed logs to /var/log |
| -c /etc/snort/snort.conf | The path to our pulledpork.conf file |

Run the following command:

```
sudo /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l
```

PulledPork should finish with output similar to the below (showing the new rules downloaded, in the example below there are over 26,000 new rules downloaded). You can ignore warnings about not running inline, since that doesn't apply to our configuration:

```
(...)
Rule Stats...
        New:-------26351
        Deleted:---0
        Enabled Rules:----8836
        Dropped Rules:----0
        Disabled Rules:---17515
        Total Rules:------26351
IP Blacklist Stats...
        Total IPs:-----9374

Done
user@snortserver:~$
```

When PulledPork completes successfully as above, You should now see `snort.rules` in `/etc/snort/rules/`.

Pulled Pork combines all the rules into one file: `/etc/snort/rules/snort.rules`. You need to make sure to add the line: `include $RULE_PATH/snort.rules` to the `snort.conf file`, or the PulledPork rules will never be read into memory when Snort starts.

Edit `/etc/snort/snort.conf`, and add to the end of the file (or at line 548 if you want to keep it in a logical place):

```
include $RULE_PATH/snort.rules
```

Since we've modified the Snort configuration file (via the loaded rules file), we should test the Snort configuration file. This will also check the new `snort.rules` file that PulledPork created:

```
sudo snort -T -c /etc/snort/snort.conf -i eth0
```

You can ignore warnings about flowbits not being checked, as well GID duplicate warnings.

Once that is successful, we want to set PulledPork to run daily. To do this, we add the PulledPork script to root's crontab:

```
sudo crontab -e
```

Append the follwoing line in crontab:

```
01 04 * * * /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l
```

Note: Snort needs to be reloaded to see the new rules. This can be done with `kill -SIGHUP <snort pid>`, or you can restart the snort service (once that's created below).

Additional note about shared object rules: In addition to regular rules, The above section will download Shared object rules. Shared object rules are also known as "Shared Object rules", "SO rules", "pre-compiled rules", or "Shared Objects". These are detection rules that are written in the Shared Object rule language, which is similar to C.
These rules are pre-compiled by the provider of the rules, and allow for more complicated rules, and allow for obfuscation of rules (say to detect attacks that haven't been patched yet, but the vendor wants to allow detection without revealing the vulnerability). These rules are compiled by the vendor for specific systems. One of these systems is Ubuntu 12, and luckily these rules also work on Ubuntu 14 and 15.

16 Creating Startup Scripts

We want to create startup scripts for Snort and Barnyard2 that will launch the services on system startup. Ubuntu 15 uses the systemd init system, while previous versions of Ubuntu use the Upstart system. If you are installing Snort on Ubuntu 12 or 14, go to the next section. If you are installing Snort on Ubuntu 15, skip the next section and go to systemD Startup Script - Ubuntu 15.

16.1   Upstart Startup Script - Ubuntu 12 and 14

We will use Upstart rather than SystemV init scrips to run both Snort and Barnyard2. First we need to create the Snort startup script:

```
sudo vi /etc/init/snort.conf
```

With the following content (note that we are using the same flags as when we tested above, except for the addition of the -D flag, which tells Snort to run as a daemon). Remember to change eth0 to the interface you want to listen on:

```
description "Snort NIDS Service"
stop on runlevel [!2345]
start on runlevel [2345]
script
     exec /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth0 -D
end script
```

Now make the script executable, and tell Upstart that the script exists, and then verify that it is installed:

```
user@snortserver:~$ sudo chmod +x /etc/init/snort.conf
user@snortserver:~$ initctl list | grep snort
snort stop/waiting
user@snortserver:~$
```

Do the same for our Barnyard2 script (note that the exec command should be one one line). We will add two flags here: -D to run as a daemon, and -a /var/log/snort/archived_logs, this will move logs that Barnyard2 has processed to the /var/log/snort/archived/ folder.

```
sudo vi /etc/init/barnyard2.conf
```

With the following content. Note that the Exec line (between script and end script) should be a single line:

```
description "Barnyard2 service"
stop on runlevel [!2345]
start on runlevel [2345]
script
    exec /usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort
    /barnyard2.waldo -g snort -u snort -D -a /var/log/snort/archived_logs
end script
```

Make the script executable and check to see that it installed correctly:

```
user@snortserver:~$ sudo chmod +x /etc/init/barnyard2.conf
user@snortserver:~$ initctl list | grep barnyard
barnyard2 stop/waiting
user@snortserver:~$
```

Reboot the computer and check that both services are started:

```
user@snortserver:~$ service snort status
snort start/running, process 1116
user@snortserver:~$ service barnyard2 status
barnyard2 start/running, process 1109
user@snortserver:~$
```

If Barnyard2 does not startup, you may need to delete then re-create the Snort database. Follow the instructions in Apendix: Troubleshooting Barnyard2 if this is needed.

Skip the next section (since you aren't installing systemD daemons) and go to Snorby - A Web GUI for Snort.

16.2   systemD Startup Script - Ubuntu 15

Ubuntu 15 has moved to systemD for services / daemons. For more information about creating and managing systemD servcies, please see this excellent article.

To create the Snort systemD service, use an editor to create a service file:

```
sudo vi /lib/systemd/system/snort.service
```

with the following content (change eth0 if different on your system):

```
[Unit]
Description=Snort NIDS Daemon
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth0

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

Now we tell systemD that the service should be started at boot:

```
sudo systemctl enable snort
```

finally, we want to start the service:

```
sudo systemctl start snort
```

to check that the service is running:

```
systemctl status snort
```

Next, create the Barnyard2 systemd service. We will add two flags here: `-D` to run as a daemon, and
`-a /var/log/snort/archived_logs`, this will move logs that Barnyard2 has processed to the
`/var/log/snort/archived/` folder. Use an editor to create a service file:

```
sudo vi /lib/systemd/system/barnyard2.service
```

with the following content (the exec content line should be one line, through `...archived_logs`):

```
[Unit]
Description=Barnyard2 Daemon
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -q -w /var/log/
     snort/barnyard2.waldo -g snort -u snort -D -a /var/log/snort/archived_logs

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

Now we tell systemD that the service should be started at boot:

```
sudo systemctl enable barnyard2
```

finally, we want to start the service:

```
sudo systemctl start barnyard2
```

to check that the service is running:

```
systemctl status barnyard2
```

Reboot and verify that both services start correctly.

17 Snorby - A Web GUI for Snort

Snorby is a web GUI for Snort that uses Ruby on Rails to provide a Web-2.0 interface. An alternative
to Snorby is BASE. BASE has a much simpler interface than Snorby, and is still very popular. Either
application will work, although I have chosen Snorby for this install guide. If you would rather use BASE,I
have instructions for installing BASE on my blog.

Another GUI option would be Sguil, or to forward the alerts to a SIEM like Splunk.

Because of differences between the various versions of Ubuntu, I have broken this installation section into
the next three sections, one for each version of Ubuntu. Please follow one of the links below for your Ubuntu
distribution:
- Install Snorby 2.6.2 on Ubuntu 12.
- Install Snorby 2.6.2 on Ubuntu 14.
- Install Snorby 2.6.2 on Ubuntu 15.

17.1   Install Snorby 2.6.2 on Ubuntu 12

Install the Snorby Pre-requisites:

```
sudo apt-get install -y imagemagick apache2 libyaml-dev libxml2-dev libxslt-dev git ruby1.9.3
```

Snorby installs a number of Ruby gems. To speed up their installation, run the following two commands to prevent the install of documentation when gems are installed:

```
echo "gem: --no-rdoc --no-ri" >  ~/.gemrc
sudo sh -c "echo gem: --no-rdoc --no-ri > /etc/gemrc"
```

Install the gems required for management and installation:

```
# These gems will also install other required gems
sudo gem install wkhtmltopdf
sudo gem install bundler
sudo gem install rails
sudo gem install rake --version=0.9.2
```

Download the 2.6.2 version of Snorby and move it to your webserver directory:

```
cd ~/snort_src/
wget https://github.com/Snorby/snorby/archive/v2.6.2.tar.gz -O snorby-2.6.2.tar.gz
tar xzvf snorby-2.6.2.tar.gz
sudo cp -r ./snorby-2.6.2/ /var/www/snorby/
```

Install all of the Snorby pre-requisites. Ignore warnings about running bundle as root. If you get connection errors when trying to download gems, just re-run the command until it succeeds.

```
cd /var/www/snorby
sudo bundle install
```

Snorby uses `database.yml` to tell it how to connect to the MySQL server. We will copy the example file to the correct location and edit it with our credentials:

```
sudo cp /var/www/snorby/config/database.yml.example /var/www/snorby/config/database.yml
sudo vi /var/www/snorby/config/database.yml
```

You need to change the password field to reflect the MySQL root password you set when installing MySQL (MySqlROOTpassword). Note that we will change this later after Snorby has setup the databases it needs to use a lower-priviledge MySQL account. The begining of the file should look like this after editing:

```
# Snorby Database Configuration
#
# Please set your database password/user below
# NOTE: Indentation is important.
#
snorby: &snorby
  adapter: mysql
  username: root
  password: "MySqlROOTpassword" # Example: password: "s3cr3tsauce"
  host: localhost

development:
... and so on
```

Now we need to create the Snorby configuration file (copied from it's example file), and update it to point to the correct version of wkhtmlpdf:

```
sudo cp /var/www/snorby/config/snorby_config.yml.example /var/www/snorby/config/snorby_config.yml
sudo sed -i s/"\/usr\/local\/bin\/wkhtmltopdf"/"\/usr\/bin\/wkhtmltopdf"/g \
        /var/www/snorby/config/snorby_config.yml
```

Now we want to install Snorby. The below command will download the necessary gems and will create a new database called Snorby for use. This can take some time to complete. You can ignore errors about "Jammit Warning: Asset compression disabled – Java unavailable.".

```
cd /var/www/snorby
sudo bundle exec rake snorby:setup
```

Now we want to edit the MySQL Snorby database to grant access to a lower privilidged user (we don't want the Snorby application using the root password to interface with the database). Run the following commands to create a new MySQL snorby user with password PASSWORD123. You will be prompted for your MySQL root password (MySqlROOTpassword) after the first command:

```
$ mysql -u root -p
myslq> create user 'snorby'@'localhost' IDENTIFIED BY 'PASSWORD123';
myslq> grant all privileges on snorby.* to 'snorby'@'localhost' with grant option;
myslq> flush privileges;
myslq> exit
```

Now that we've created a new MySQL snorby user and password, edit Snorby's database.yml to use the new account:

```
sudo vi /var/www/snorby/config/database.yml
```

The file should now look like this (note the changes to lines 8 and 9):

```
# Snorby Database Configuration
#
# Please set your database password/user below
# NOTE: Indentation is important.
#
snorby: &snorby
  adapter: mysql
  username: snorby
  password: "PASSWORD123" # Example: password: "s3cr3tsauce"
  host: localhost

development:
  database: snorby
  <<: *snorby

test:
  database: snorby
  <<: *snorby

production:
  database: snorby
  <<: *snorby
```

Now we are ready to test Snorby. Run Snorby with:

```
cd /var/www/snorby/
sudo bundle exec rails server -e production
```

This will start Snorby and will be available on port 3000.

Navigate to http://<ip_of_snorby_server>:3000 and you should see the logon screen. Don't log in at this time as we are only testing that the software runs. Use ctrl-c to stop the Snorby server.

We will use Phusion Passenger, an application server module for Apache to launch Snorby. First install pre-requisites:

```
sudo apt-get install -y libcurl4-openssl-dev apache2-threaded-dev libaprutil1-dev libapr1-dev
```

Install the Passenger gem and the apache module (we don't install the Ubuntu repository version of Phusion Passenger because it doesn't work well).

```
sudo gem install passenger
sudo passenger-install-apache2-module
```

The Phusion Passenger install wizard will start. Un-check the Python language support (we only need Ruby support) using the arrows and space bar, then use enter to continue through the menu options.

After compiling software, the wizard will finally tell you to copy some text to your Apache configuration file. We don't want to do that because Apache now uses separate files for modules. We do want the information that is printed, we will just use it slightly differently. Copy the six lines of text that are shown, as you'll need them. Hit enter twice to exit the wizard. My install showed the following (yours may be different):

```
LoadModule passenger_module /var/lib/gems/1.9.1/gems/passenger-5.0.21/buildout/apache2/mod_passenger.so
<IfModule mod_passenger.c>
  PassengerRoot /var/lib/gems/1.9.1/gems/passenger-5.0.21
  PassengerDefaultRuby /usr/bin/ruby1.9.1
</IfModule>
```

The first line tells Apache the path to the shared object library to load the Phusion passenger module. We want to create a new file for this line. Create this file:

```
sudo vi /etc/apache2/mods-available/passenger.load
```

And paste the first line into that file. In my case, I pasted:

```
LoadModule passenger_module /var/lib/gems/1.9.1/gems/passenger-5.0.21/buildout/apache2/mod_passenger.so
```

The final 4 lines specify the configuration for Phusion Passenger. Create the correct file as follows:

```
sudo vi /etc/apache2/mods-available/passenger.conf
```

And paste the two content lines in. You do not need the <IfModule>tags In my case, I pasted:

```
PassengerRoot /var/lib/gems/1.9.1/gems/passenger-5.0.21
PassengerDefaultRuby /usr/bin/ruby1.9.1
```

Note: yes, the lines above say ruby1.9.1, and we did install ruby 1.9.3. Ubuntu 12 does some folder redirection that makes this happen, but it doesn't cause any issues.

Enable the Passenger module:

```
sudo a2enmod passenger
sudo service apache2 restart
```

and then verify that it loaded (look for Passenger in the output):

```
apache2ctl -t -D DUMP_MODULES
```

Now we need to create a website for Snorby:

```
sudo vi /etc/apache2/sites-available/snorby.conf
```

Input the following into that file:

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName snorby.sublimerobots.com
        DocumentRoot /var/www/snorby/public
        <Directory "/var/www/snorby/public">
                AllowOverride all
                Order deny,allow
                Allow from all
                Options -MultiViews
        </Directory>
</VirtualHost>
```

Now enable the new site, disable the default site, and reload Apache to see the new configurations:

```
cd /etc/apache2/sites-available/
sudo a2ensite snorby.conf
sudo service apache2 reload

cd /etc/apache2/sites-enabled
sudo a2dissite 000-default
sudo service apache2 reload
```

Now we need to tell Barnyard2 to output events to the Snorby database that we created above.

```
sudo vi /etc/snort/barnyard2.conf
```

Append at the end off the file:

```
output database: log, mysql, user=snorby password=PASSWORD123 dbname=snorby host=localhost sensor_name=sensor1
```

We can disable the other output file that you created during the Barnyard2 testing by deleting the previous line (or putting a hash in front of it to disable it).

```
# output database: log, mysql, user=snort password=MySqlSNORTpassword dbname=snort host=localhost)
```

Restart Barnyard2 to load the new configuration:

```
sudo service barnyard2 restart
```

Snorby needs one service running for database maintenance (a Snorby worker daemon). We will create an Upstart daemon for this task.

First we need to create the startup script:

```
sudo vi /etc/init/snorby_worker.conf
```

with the following content:

```
description "Snorby Delayed Job"
stop on runlevel [!2345]
start on runlevel [2345]
chdir /var/www/snorby

script
        exec /usr/bin/ruby script/delayed_job start
end script
```

Now make the script executable, and tell Upstart that the script exists, and then verify that it installed correctly:

```
user@snortserver:~$ sudo chmod +x /etc/init/snorby_worker.conf
user@snortserver:~$ initctl list | grep snorby_worker
snorby_worker stop/waiting
user@snortserver:~$
```

Note that this daemon will often list as stop/waiting, and that is ok, because of how it works. You can check the worker job status use the web interface (look under Administration –>Worker and Job Queue).

To log into the web interface: open a web browser and navigate to http://<ip_of_snorby_server>. you don't need to enter the port number, as it is listening on port 80 now.

The default login information is:

```
        E-mail:   snorby@snorby.org
        Password: snorby
```

(current Snorby documentation lists the email as example.com, not snorby.org. That is incorrect for the 2.6.2 version of Snorby, where the username remains snorby@snorby.org)

It can take some time (a minute or two) between when alerts are generated and when they show up in the Snorby dashbord. To verify that alerts are being written to the Snorby database, generate some events (using the ping rule as before), then run the following command, using the Snorby MySql password (PASSWORD123):

```
mysql -u snorby -p -D snorby -e "select count(*) from event"
```

If you have issues, there is a good chance they are related to Barnyard2. Please see Apendix: Troubleshooting Barnyard2.

If everything is working, go to the next section: Where To Go From Here.

17.2   Install Snorby 2.6.2 on Ubuntu 14

Install the Snorby Pre-requisites:

```
sudo apt-get install -y imagemagick apache2 libyaml-dev libxml2-dev libxslt-dev git ruby1.9.3
```

Snorby installs a number of Ruby gems. To speed up their installation, run the following two commands to prevent the install of documentation when gems are installed:

```
echo "gem: --no-rdoc --no-ri" >  ~/.gemrc
sudo sh -c "echo gem: --no-rdoc --no-ri > /etc/gemrc"
```

Install the gems required for management and installation:

```
# These gems will also install other required gems
sudo gem install wkhtmltopdf
sudo gem install bundler
sudo gem install rails
sudo gem install rake --version=0.9.2
```

Download the 2.6.2 version of Snorby and move it to your webserver directory:

```
cd ~/snort_src/
wget https://github.com/Snorby/snorby/archive/v2.6.2.tar.gz -O snorby-2.6.2.tar.gz
tar xzvf snorby-2.6.2.tar.gz
sudo cp -r ./snorby-2.6.2/ /var/www/html/snorby/
```

Install all of the Snorby pre-requisites. Ignore warnings about running bundle as root. If you get connection errors when trying to download gems, just re-run the command until it succeeds.

```
cd /var/www/html/snorby
sudo bundle install
```

Snorby uses `database.yml` to tell it how to connect to the MySQL server. We will copy the example file to the correct location and edit it with our credentials:

```
sudo cp /var/www/html/snorby/config/database.yml.example /var/www/html/snorby/config/database.yml
sudo vi /var/www/html/snorby/config/database.yml
```

You need to change the password field to reflect the MySQL root password you set when installing MySQL (MySqlROOTpassword). Note that we will change this later after Snorby has setup the databases it needs to use a lower-priviledge MySQL account. The file should look like this after editing:

```
# Snorby Database Configuration
#
# Please set your database password/user below
# NOTE: Indentation is important.
#
snorby: &snorby
  adapter: mysql
  username: root
  password: "MySqlROOTpassword" # Example: password: "s3cr3tsauce"
  host: localhost

development:
  database: snorby
  <<: *snorby

test:
  database: snorby
  <<: *snorby

production:
  database: snorby
  <<: *snorby
```

Now we need to create the Snorby configuration file (copied from it's example file), and update it to point to the correct version of wkhtmlpdf:

```
sudo cp /var/www/html/snorby/config/snorby_config.yml.example /var/www/html/snorby/config/snorby_config.yml
sudo sed -i s/"\/usr\/local\/bin\/wkhtmltopdf"/"\/usr\/bin\/wkhtmltopdf"/g \
        /var/www/html/snorby/config/snorby_config.yml
```

Now we want to install Snorby. The below command will download the necessary gems and will create a new database called Snorby for use. This can take some time to complete. You can ignore errors about "Jammit Warning: Asset compression disabled – Java unavailable.".

```
cd /var/www/html/snorby
sudo bundle exec rake snorby:setup
```

Now we want to edit the MySQL Snorby database to grant access to a lower privilidged user (we don't want the Snorby application using the root password to interface with the database). Run the following commands to create a new MySQL snorby user with password PASSWORD123. You will be prompted for your MySQL root password (MySqlROOTpassword) after the first command:

```
$ mysql -u root -p
myslq> create user 'snorby'@'localhost' IDENTIFIED BY 'PASSWORD123';
myslq> grant all privileges on snorby.* to 'snorby'@'localhost' with grant option;
myslq> flush privileges;
myslq> exit
```

Now that we've created a new MySQL snorby user and password, edit Snorby's database.yml to use the new account:

```
sudo vi /var/www/html/snorby/config/database.yml
```

The file should now look like this (note the changes to lines 8 and 9):

```
# Snorby Database Configuration
#
# Please set your database password/user below
# NOTE: Indentation is important.
#
snorby: &snorby
  adapter: mysql
  username: snorby
  password: "PASSWORD123" # Example: password: "s3cr3tsauce"
  host: localhost

development:
  database: snorby
  <<: *snorby

test:
  database: snorby
  <<: *snorby

production:
  database: snorby
  <<: *snorby
```

Now we are ready to test Snorby. Run Snorby with:

```
cd /var/www/html/snorby/
sudo bundle exec rails server -e production
```

This will start Snorby and will be available on port 3000.
Navigate to http://<ip_of_snorby_server>:3000 and you should see the logon screen. Don't log in at this time as we are only testing that the software runs. Use ctrl-c to stop the Snorby server.

We will use Phusion Passenger, an application server module for Apache to launch Snorby. First install pre-requisites:

```
sudo apt-get install -y libcurl4-openssl-dev apache2-threaded-dev libaprutil1-dev libapr1-dev
```

Install the Passenger gem and the apache module (we don't install the Ubuntu repository version of Phusion Passenger because it doesn't work well).

```
sudo gem install passenger
sudo passenger-install-apache2-module
```

The Phusion Passenger install wizard will start. Un-check the Python language support (we only need Ruby support) using the arrows and space bar, then use enter to continue through the menu options.

After compiling software, the wizard will finally tell you to copy some text to your Apache configuration file. We don't want to do that because Apache now uses separate files for modules. We do want the information that is printed, we will just use it slightly differently. Copy the six lines of text that are shown, as you'll need them. Hit enter twice to exit the wizard. My install showed the following (yours may be different):

```
LoadModule passenger_module /var/lib/gems/1.9.1/gems/passenger-5.0.21/buildout/apache2/mod_passenger.so
<IfModule mod_passenger.c>
  PassengerRoot /var/lib/gems/1.9.1/gems/passenger-5.0.21
  PassengerDefaultRuby /usr/bin/ruby1.9.1
</IfModule>
```

The first line tells Apache the path to the shared object library to load the Phusion passenger module. We want to create a new file for this line. Create this file:

```
sudo vi /etc/apache2/mods-available/passenger.load
```

And paste the first line into that file. In my case, I pasted:

```
LoadModule passenger_module /var/lib/gems/1.9.1/gems/passenger-5.0.21/buildout/apache2/mod_passenger.so
```

The final 4 lines specify the configuration for Phusion Passenger. Create the correct file as follows:

```
sudo vi /etc/apache2/mods-available/passenger.conf
```

And paste the two content lines in. You do not need the <IfModule>tags In my case, I pasted:

```
PassengerRoot /var/lib/gems/1.9.1/gems/passenger-5.0.21
PassengerDefaultRuby /usr/bin/ruby1.9.1
```

Note: yes, the lines above say ruby1.9.1, and we did install ruby 1.9.3. Ubuntu 12 does some folder redirection that makes this happen, but it doesn't cause any issues.

Enable the Passenger module:

```
sudo a2enmod passenger
sudo service apache2 restart
```

and then verify that it loaded (look for Passenger in the output):

```
apache2ctl -t -D DUMP_MODULES
```

Now we need to create a website for Snorby:

```
sudo vi /etc/apache2/sites-available/snorby.conf
```

Input the following into that file:

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName snorby.sublimerobots.com
        DocumentRoot /var/www/html/snorby/public
        <Directory "/var/www/html/snorby/public">
                AllowOverride all
                Order deny,allow
                Allow from all
                Options -MultiViews
        </Directory>
</VirtualHost>
```

Now enable the new site, disable the default site, and reload Apache to see the new configurations:

```
cd /etc/apache2/sites-available/
sudo a2ensite snorby.conf
sudo service apache2 reload

cd /etc/apache2/sites-enabled
sudo a2dissite 000-default
sudo service apache2 reload
```

Now we need to tell Barnyard2 to output events to the Snorby database that we created above.

```
sudo vi /etc/snort/barnyard2.conf
```

Append at the end off the file:

```
output database: log, mysql, user=snorby password=PASSWORD123 dbname=snorby host=localhost sensor_name=sensor1
```

We can disable the other output file that you created during the Barnyard2 testing by deleting the previous line (or putting a hash in front of it to disable it).

```
# output database: log, mysql, user=snort password=MySqlSNORTpassword dbname=snort host=localhost)
```

Restart Barnyard2 to load the new configuration:

```
sudo service barnyard2 restart
```

Snorby needs one service running for database maintenance (a Snorby worker daemon). We will create an Upstart daemon for this task.

First we need to create the startup script:

```
sudo vi /etc/init/snorby_worker.conf
```

with the following content:

```
description "Snorby Delayed Job"
stop on runlevel [!2345]
start on runlevel [2345]
chdir /var/www/html/snorby
script
        exec /usr/bin/ruby script/delayed_job start
end script
```

Now make the script executable, and tell Upstart that the script exists, and then verify that it installed correctly:

```
user@snortserver:~$ sudo chmod +x /etc/init/snorby_worker.conf
user@snortserver:~$ initctl list | grep snorby_worker
snorby_worker stop/waiting
user@snortserver:~$
```

Note that this daemon will often list as stop/waiting, and that is ok, because of how it works. You can check the worker job status use the web interface (look under Administration –>Worker and Job Queue).

To log into the web interface: open a web browser and navigate to http://<ip_of_snorby_server>. you don't need to enter the port number, as it is listening on port 80 now.

The default login information is:

```
        E-mail:   snorby@snorby.org
        Password: snorby
```

(current Snorby documentation lists the email as example.com, not snorby.org. That is incorrect for the 2.6.2 version of Snorby, where the username remains snorby@snorby.org)

It can take some time (a minute or two) between when alerts are generated and when they show up in the Snorby dashbord. To verify that alerts are being written to the snorby database, generate some events (using the ping rule as before), then run the following command, using the snorby MySql password (PASS-WORD123):

```
mysql -u snorby -p -D snorby -e "select count(*) from event"
```

If you have issues, there is a good chance they are related to Barnyard2. Please see Apendix: Troubleshooting Barnyard2.

If everything is working, go to the next section: Where To Go From Here.

17.3   Install Snorby 2.6.2 on Ubuntu 15

On Ubuntu 15, ruby 1.9 is not available in the repositories, so we need to download and install from source. We need some pre-requisites:

```
sudo apt-get install -y libgdbm-dev libncurses5-dev git-core curl zlib1g-dev build-essential \
        libssl-dev libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev \
        libcurl4-openssl-dev python-software-properties libffi-dev
```

We also need some pre-requisites for the ruby gems that we will use:

```
sudo apt-get install -y imagemagick apache2 libyaml-dev libxml2-dev libxslt-dev git libssl-dev
```

Snorby installs a number of Ruby gems. To speed up their installation, run the following two commands to prevent the install of documentation when gems are installed:

```
echo "gem: --no-rdoc --no-ri" >  ~/.gemrc
sudo sh -c "echo gem: --no-rdoc --no-ri > /etc/gemrc"
```

Download the ruby 1.9.3 source and install

```
cd ~/snort_src/
wget https://cache.ruby-lang.org/pub/ruby/1.9/ruby-1.9.3-p551.tar.gz
tar -xvzf ruby-1.9.3-p551.tar.gz
cd ruby-1.9.3-p551
./configure
make
sudo make install
```

Install the gems required for management and installation:

```
# These gems will also install other required gems
sudo gem install wkhtmltopdf
sudo gem install bundler
sudo gem install rails
sudo gem install rake --version=0.9.2
```

Download the 2.6.2 version of Snorby and move it to your webserver directory:

```
cd ~/snort_src/
wget https://github.com/Snorby/snorby/archive/v2.6.2.tar.gz -O snorby-2.6.2.tar.gz
tar xzvf snorby-2.6.2.tar.gz
sudo cp -r ./snorby-2.6.2/ /var/www/html/snorby/
```

Install all of the Snorby pre-requisites. Ignore warnings about running bundle as root. If you get connection errors when trying to download gems, just re-run the command until it succeeds.

```
cd /var/www/html/snorby
sudo bundle install
```

Snorby uses `database.yml` to tell it how to connect to the MySQL server. We will copy the example file to the correct location and edit it with our credentials:

```
sudo cp /var/www/html/snorby/config/database.yml.example /var/www/html/snorby/config/database.yml
sudo vi /var/www/html/snorby/config/database.yml
```

You need to change the password field to reflect the MySQL root password you set when installing MySQL (MySqlROOTpassword). Note that we will change this later after Snorby has setup the databases it needs to use a lower-priviledge MySQL account. The first lines of the file should look like this after editing:

```
# Snorby Database Configuration
#
# Please set your database password/user below
# NOTE: Indentation is important.
#
snorby: &snorby
  adapter: mysql
  username: root
  password: "MySqlROOTpassword" # Example: password: "s3cr3tsauce"
  host: localhost

development:
 ...and so on
```

Now we need to create the Snorby configuration file (copied from it's example file), and update it to point to the correct version of wkhtmlpdf:

```
sudo cp /var/www/html/snorby/config/snorby_config.yml.example /var/www/html/snorby/config/snorby_config.yml
sudo sed -i s/"\/usr\/local\/bin\/wkhtmltopdf"/"\/usr\/bin\/wkhtmltopdf"/g \
        /var/www/html/snorby/config/snorby_config.yml
```

Now we want to install Snorby. The below command will download the necessary gems and will create a new database called Snorby for use. This can take some time to complete. You can ignore errors about "Jammit Warning: Asset compression disabled – Java unavailable.".

```
cd /var/www/html/snorby
sudo bundle exec rake snorby:setup
```

Now we want to edit the MySQL Snorby database to grant access to a lower privilidged user (we don't want the Snorby application using the root password to interface with the database). Run the following commands to create a new MySQL snorby user with password PASSWORD123. You will be prompted for your MySQL root password (MySqlROOTpassword) after the first command:

```
$ mysql -u root -p
myslq> create user 'snorby'@'localhost' IDENTIFIED BY 'PASSWORD123';
myslq> grant all privileges on snorby.* to 'snorby'@'localhost' with grant option;
myslq> flush privileges;
myslq> exit
```

Now that we've created a new MySQL snorby user and password, edit Snorby's database.yml to use the new account:

```
sudo vi /var/www/html/snorby/config/database.yml
```

The file should now look like this (note the changes to lines 8 and 9):

```
# Snorby Database Configuration
#
# Please set your database password/user below
# NOTE: Indentation is important.
#
snorby: &snorby
  adapter: mysql
  username: snorby
  password: "PASSWORD123" # Example: password: "s3cr3tsauce"
  host: localhost

development:
  database: snorby
  <<: *snorby

test:
  database: snorby
  <<: *snorby

production:
  database: snorby
  <<: *snorby
```

Now we are ready to test Snorby. Run Snorby with:

```
cd /var/www/html/snorby/
sudo bundle exec rails server -e production
```

This will start Snorby and will be available on port 3000.
Navigate to http://<ip_of_snorby_server>:3000 and you should see the logon screen. Don't log in at this time as we are only testing that the software runs. Use ctrl-c to stop the Snorby server.

31

We will use Phusion Passenger, an application server module for Apache to launch Snorby. First install pre-requisites:

```
sudo apt-get install -y libcurl4-openssl-dev apache2-threaded-dev libaprutil1-dev libapr1-dev
```

Install the Passenger gem and the apache module (we don't install the Ubuntu repository version of Phusion Passenger because it doesn't work well).

```
sudo gem install passenger
sudo passenger-install-apache2-module
```

The Phusion Passenger install wizard will start. Un-check the Python language support (we only need Ruby support) using the arrows and space bar, then use enter to continue through the menu options.

After compiling software, the wizard will finally tell you to copy some text to your Apache configuration file. We don't want to do that because Apache now uses separate files for modules. We do want the information that is printed, we will just use it slightly differently. Copy the six lines of text that are shown, as you'll need them. Hit enter twice to exit the wizard. My install showed the following (yours may be different):

```
  LoadModule passenger_module /usr/local/lib/ruby/gems/1.9.1/gems/passenger-5.0.21/buildout/apache2/
    mod_passenger.so
  <IfModule mod_passenger.c>
    PassengerRoot /usr/local/lib/ruby/gems/1.9.1/gems/passenger-5.0.21
    PassengerDefaultRuby /usr/local/bin/ruby
  </IfModule>
```

The first line tells Apache the path to the shared object library to load the Phusion passenger module. We want to create a new file for this line. Create this file:

```
sudo vi /etc/apache2/mods-available/passenger.load
```

And paste the first line into that file. In my case, I pasted:

```
LoadModule passenger_module /usr/local/lib/ruby/gems/1.9.1/gems/passenger-5.0.21/buildout/apache2/
    mod_passenger.so
```

The final 4 lines specify the configuration for Phusion Passenger. Create the correct file as follows:

```
sudo vi /etc/apache2/mods-available/passenger.conf
```

And paste the two content lines in. You do not need the <IfModule>tags In my case, I pasted:

```
PassengerRoot /usr/local/lib/ruby/gems/1.9.1/gems/passenger-5.0.21
PassengerDefaultRuby /usr/local/bin/ruby
```

Note: yes, the lines above say ruby1.9.1, and we did install ruby 1.9.3. Ubuntu 12 does some folder redirection that makes this happen, but it doesn't cause any issues.

Enable the Passenger module:

```
sudo a2enmod passenger
sudo service apache2 restart
```

and then verify that it loaded (look for Passenger in the output):

```
apache2ctl -t -D DUMP_MODULES
```

Now we need to create a website for Snorby:

```
sudo vi /etc/apache2/sites-available/snorby.conf
```

Input the following into that file:

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName snorby.sublimerobots.com
        DocumentRoot /var/www/html/snorby/public
        <Directory "/var/www/html/snorby/public">
                AllowOverride all
                Order deny,allow
                Allow from all
                Options -MultiViews
        </Directory>
</VirtualHost>
```

Now enable the new site, disable the default site, and reload Apache to see the new configurations:

```
cd /etc/apache2/sites-available/
sudo a2ensite snorby.conf
sudo service apache2 reload

cd /etc/apache2/sites-enabled
sudo a2dissite 000-default
sudo service apache2 reload
```

Now we need to tell Barnyard2 to output events to the Snorby database that we created above.

```
sudo vi /etc/snort/barnyard2.conf
```

Append at the end off the file:

```
output database: log, mysql, user=snorby password=PASSWORD123 dbname=snorby host=localhost sensor_name=sensor1
```

We can disable the other output file that you created during the Barnyard2 testing by deleting the previous line (or putting a hash in front of it to disable it).

```
# output database: log, mysql, user=snort password=MySqlSNORTpassword dbname=snort host=localhost)
```

Restart Barnyard2 to load the new configuration:

```
sudo service barnyard2 restart
```

Snorby needs one service running for database maintenance (a Snorby worker daemon). We will create a systemD daemon for this task.

First we need to create the service file:

```
sudo vi /lib/systemd/system/snorby_worker.service
```

with the following content:

```
[Unit]
Description=Snorby Worker Daemon
Requires=apache2.service
After=syslog.target network.target apache2.service

[Service]
Type=forking
WorkingDirectory=/var/www/html/snorby
ExecStart=/usr/local/bin/ruby script/delayed_job start

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

Now make the script executable, and tell systemD that the script exists, and then verify that it installed correctly:

```
sudo systemctl enable snorby_worker
systemctl status snorby_worker.service
```

We do not want to start the Snorby worker daemon at this time because an instance of the worker is already running (from when you installed Snorby). You can verify this by logging into the web interface using the instructions below. In the Snorby web interface, look for the snorby worker status under **Administration −>Worker and Job Queue**).

To log into the web interface: open a web browser and navigate to http://<ip_of_snorby_server>. you don't need to enter the port number, as it is listening on port 80 now.

The default login information is:

```
        E-mail:   snorby@snorby.org
        Password: snorby
```

(current Snorby documentation lists the email as example.com, not snorby.org. That is incorrect for the 2.6.2 version of Snorby, where the username remains snorby@snorby.org)

It can take some time (a minute or two) between when alerts are generated and when they show up in the Snorby dashboard. To verify that alerts are being written to the snorby database, generate some events (using the ping rule as before), then run the following command, using the snorby MySql password (PASSWORD123):

```
mysql -u snorby -p -D snorby -e "select count(*) from event"
```

If events don't show, try rebooting the system and checking again.

Reboot the computer and verify that the worker process has restarted properly in the web interface (look under **Administration −>Worker and Job Queue**). If it doesn't show as running, check the service status and the service log:

```
systemctl status snorby_worker.service

journalctl -u snorby_service
```

18   Where To Go From Here

I hope this guide has been helpful to you. Please feel free to provide feedback, both issues you experienced and recommendations that you have. The goal of this guide was not just for you to create a Snort NIDS, but to understand how all the parts work together, and get a deeper understanding of all the components, so that you can troubleshoot and modify your Snort NIDS with confidence.

Capturing More Traffic With Snort:

You will probably want to configure your network infrastructure to mirror traffic meant for other hosts to your Snort sensor. This configuration is dependent on what network equipment you are using. If you are running Snort as a Virtual Machine on a VMware ESXi server, you can configure promiscuous mode for ESXi by following the instructions in Appendix: ESXi and Snort in Promiscuous Mode.

For different network infrastructure, you will need to do a little research to configure network mirroring for your Snort server. Cisco calls this a span port, but most other vendors call this Port Mirroring. Instructions for Mikrotik (a linux based switch and router product that i like). If you run DD-WRT, it can be configured with iptables, like any linux based system. If you have network equipment not listed above, any search engine should point you towards a solution, if one exists. Note that many consumer switches will not have the ability to mirror ports.

More Advanced Snort Configuration

Snort has the ability to do much more than weve covered in this set of articles. Hopefully youve learned enough through this setup that you will be able to implement more advanced configurations and make Snort work for you. Some things that Snort is capable of:

- Network Intrusion Prevention System (IPS)

- Multiple remote Snort sensors, for example on different subnets.

- The documentation section of the Snort website has a number of useful articles about more advanced things you can do with Snort.

Recommended Reading

- *Snort IDS and IPS Toolkit* (Jay Beale's Open Source Security) This is a good book for understanding how Snort works under the hood. It is a little old, but is still relevant and very detailed.

- *Snort 2.1 Intrusion Detection, 2nd Edition* Another book by Jay Beale, again this is an excellent book on the Snort engine and architecture. The supporting tools he references are out of date and no longer supported, but the rest of the book is excellent.

- *Snort Cookbook* This book is very helpful in showing how Snort can be run to meet specific needs (using recipes that describe specific situations).

- *Applied Network Security Monitoring: Collection, Detection, and Analysis* I havent read this book, but it is well reviewed, and covers NIDS at a much higher level than the other two books.

Feedback I would love to get feedback from you about this guide. Recommendations, issues, or ideas, please email me: Noah@SublimeRobots.com.

A   Appendix: ESXi and Snort in Promiscuous Mode

Often you want your Snort NIDS to listen on an adapter that receives all traffic for a switch. VMware calls this "Promiscuous Mode", while Cisco calls this a "Mirror Port". To configure your ESXi server to mirror all traffic to an interface on a Virtual Machine (such as the interface for our Snort VM), follow the steps below, from VMware's website[7]:

1. Log into the ESXi/ESX host or vCenter Server using the ESXi Client.
2. Select the ESXi/ESX host (the VMware Server) in the inventory.
3. Click the Configuration tab.
4. In the Hardware section, click Networking.
5. Click Properties of the virtual switch (the switch that Snort has its listening interface on) for which you want to enable promiscuous mode.
6. Select the virtual switch or portgroup you wish to modify and click Edit.
7. Click the Security tab.
8. From the Promiscuous Mode dropdown menu, click Accept.

Once VMware is configured to permit promiscuous mode, you then need to configure the interface that Snort is listening on for promiscuous mode. to do this, edit `/etc/network/interfaces`:

```
sudo vi /etc/network/interfaces
```

and make modifications similar to the following, depending on the configuration of your system:

```
# The primary network interface
auto eth0
iface eth0 inet dhcp

# Interface that Snort listens on
auto eth1
iface eth1 inet manual
        up ifconfig $IFACE 0.0.0.0 up
        up ip link set $IFACE promisc on
        down ip link set $IFACE promisc off
        down ifconfig $IFACE down
```

In the above example, Snort will listen on `eth1` (remember that this also has to be changed in the Snort daemon script (the interface referenced by the `-i` flag in `/etc/init/snort.conf`). We choose not to set an IP addresson the interface that Snort will listen on, since this helps to protect the system from exploits. Management in the above example will be through `eth0` which is configured for DHCP. The command `up ip link set $IFACE promisc on` is what configures the interface for promiscuous mode, and is how the system knows to process all traffic the interface sees, not just traffic that is specifically for the adapter.

To test this configuration, restart networking (or restart the system) and ensure that Snort has started, and is listening on the correct interface. Ping between two hosts on the subnet (two hosts that are not the Snort server) and you should see the events logged.

An easy way to see if this is working is to stop the Snort Daemon (with `sudo service snort stop`), then run the following (change the interface as needed):

```
$ sudo /usr/local/bin/snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth1
```

When you ping between the two hosts that aren't the Snort server, but which are on the same subnet as the Snort server, you should see the events written to the screen. Use `ctr-c` to stop Snort.

---

[7]http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1004099

B Apendix: Installing Snort Rules Manually

If you just want to test Snort manually, and want to use the rules from snort without setting up PulledPork, follow the instructions below. You will need a Oinkcode (free with an account from snort.org)

We need to un-comment all the #include lines in snort.conf, as the downloaded rules will be a series of rule files, rather than the one that PulledPork creates:

```
sudo sed -i 's/\#include \$RULE\_PATH/include \$RULE\_PATH/' /etc/snort/snort.conf
```

Download the rules, replacing <oinkcode>with your personal Snort code. you might also want to get a newer version of the rules (the example below points to the 2.9.8.0 version of the rules):

```
cd ~/snort_src
wget https://www.snort.org/reg-rules/snortrules-snapshot-2956.tar.gz/<SNORTCODE> -O snortrules-snapshot-2980.
    tar.gz
sudo tar xvfvz snortrules-snapshot-2980.tar.gz -C /etc/snort
```

Move all new files from **/etc/snort/etc** to **/etc/snort** (and get rid of **/etc/snort/etc** folder that was copied as well):

```
sudo cp ./*.conf* ../
sudo cp ./*.map ../
cd /etc/snort
sudo rm -Rf /etc/snort/etc
```

Now modify **/etc/snort/snort.conf** with any changes from the original `snort.conf`.
We want the new `snort.conf` in case it references any new rulesets.

Test the configuration file with Snort:

```
sudo snort -T -c /etc/snort/snort.conf
```

You can now run snort as you normally would (with a startup script or manually).

C   Apendix: Troubleshooting Barnyard2

If barnyard2 is having issues loading events, sometimes deleting all of snort's unified2 event logs and recreate the waldo file can help (you'll loose the events that are saved there)

To do this:

```
sudo rm /var/log/snort/*
sudo touch /var/log/snort/barnyard2.waldo
```

Other troubleshooting steps:

- Reboot the server.

- Be patient. When barnyard2 has a large number of events to process, it can take some time before they show in the database (say you accidentally ran `sudo ping -i 0.001 10.0.0.104` for a minute, generating upwards of 30,000 alerts on your snort server. This can take some time to process.

- Check that the events are in the database. To check the Snorby database:

```
mysql -u snorby -p -D snorby -e "select count(*) from event"
```

- to check the snort database if you haven't installed Snorby:

```
mysql -u snort -p -D snort -e "select count(*) from event"
```

- Check the system log

```
cat /var/log/syslog | grep barnyard
```

- Check if the services are running

```
# upstart or systemD:
service snort status
service barnyard2 status
```

- Check that the Snorby worker process is running, both in the Snorby web interface and the daemon:

```
# upstart or systemD:
service snorby_worker status

# systemD check service event log:
journalctl -u snorby_worker
```