Snort 3 on Ubuntu 14, 16, & 18

(Snort 3.0.0-a4 build 245)

Noah Dietrich
Noah@SublimeRobots.com

July 4, 2018

Contents

1 Introduction

This guide is will walk you through installing Snort3 Alpha on Ubuntu 14, 16, and 18. These instructions will not work on Ubuntu 12 (although an old guide can be found on my website).

Note: If you are installing Snort3 on Ubuntu 14 x86, you will need to install a newer version of gcc than what's available in the standard repository in order to compile Snort. I will not cover these steps for that specific platform. All other builds and platforms will work with no issues.

Please Note that Snort3 is alpha software. It does contain bugs, and new releases can have different pre-requisites than when this guide was released. This guide has been written and tested against Snort 3.0.0-a4 build 245, released on May 24, 2018. You may be able to follow this guide to install later releases of Snort, but the requirements may change.

Since Snort++ is alpha software, you should not use it in production systems. This software has been released by the Snort team to solicit feedback and for users to test.

This guide will not discuss how to configure Snort as an NIPS or NIDS, how to install additional supporting software (barnyard2, PulledPork, GUIs), how to setup network interfaces to capture data, Snort sensor design considerations, descriptions of specific preprocessors, or detailed OpenAppID use. You can read other articles I have written about these items, as well as how to configure Snort as a full NIDS or NIPS system on my website: SublimeRobots.com.

Feedback on this guide is encouraged: **Noah@SublimeRobots.com**.

You can also ask for help on the Snort distribution lists:
Snort Users
Snort Developers
Snort OpenAppID

2 Begin

First, ensure the system is up to date and has the latest list of packages:

```
sudo apt-get update && sudo apt-get dist-upgrade -y
```

We will be downloading a number of source tarballs and other source files, we want to store them in one folder:

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

Install the Snort3 prerequisites. Details of these packages can be found in the requirements section of the Snort3 Manual:

```
sudo apt-get install -y build-essential autotools-dev libdumbnet-dev libluajit-5.1-dev libpcap-dev \
        libpcre3-dev zlib1g-dev pkg-config libhwloc-dev
```

For Ubuntu 16 and 18, you can install cmake from the default repository:

```
# Ubuntu 16 and 18 (and greater) only:
sudo apt-get install -y cmake
```

For Ubuntu 14, you have to install cmake from source, because the version in the Ubuntu respository is too old to compile Snort:

```
# Ubuntu 14 only
sudo apt-get remove -y cmake
cd ~/snort_src
wget https://cmake.org/files/v3.10/cmake-3.10.3.tar.gz
tar -xzvf cmake-3.10.3.tar.gz
cd cmake-3.10.3
./bootstrap
make
sudo make install
```

Note: If you are running Ubuntu 14 x86, you must install a newer version of gcc, either from a ppa or from source. Snort will not compile properly on the x86 platform with the version of gcc in the Ubuntu repository (4.8.4). I do not cover these steps in this guide, but you can easily google this process. This does not apply to the x64 version of Ubuntu 14 which you can easily install using the steps below.

Next install the optional (but highly recommended software):

```
sudo apt-get install -y liblzma-dev openssl libssl-dev cpputest libsqlite3-dev \
        uuid-dev
```

If you want to build the latest documentation from the source tree including Snort++ Developers Guide, install the following (purely optional) packages. These packages are nearly 800 MB in size:

```
sudo apt-get install -y asciidoc dblatex source-highlight w3m
```

Since we will install Snort from the github repository, we need a few tools:

```
sudo apt-get install -y libtool git autoconf
```

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

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

If you want to run Snort in inline mode using NFQ, install the required packages (not required for IDS mode or inline mode using afpacket). If you're unsure, you should install this package.

```
sudo apt-get install -y libnetfilter-queue-dev
```

Download and install safec for runtime bounds checks on certain legacy C-library calls (this is optional but recommended):

```
cd ~/snort_src
wget https://downloads.sourceforge.net/project/safeclib/libsafec-10052013.tar.gz
tar -xzvf libsafec-10052013.tar.gz
cd libsafec-10052013
./configure
make
sudo make install
```

Download and install gperftools 2.7, google's thread-caching malloc (used in chrome). Tcmalloc is a memory allocator that's optimized for high concurrency situations which will provide better speed for the trade-off of higher memory usage. We don't want the version of tcmalloc from the repositories (version 2.5) as they don't work with Snort. Tcmalloc is optional but recommended:

```
cd ~/snort_src
wget https://github.com/gperftools/gperftools/releases/download/gperftools-2.7/gperftools-2.7.tar.gz
tar xzvf gperftools-2.7.tar.gz
cd gperftools-2.7
./configure
make
sudo make install
```

Snort3 uses Hyperscan for fast pattern matching. Hyperscan requires Ragel and the Boost headers:

Download and install Ragel:

```
cd ~/snort_src
wget http://www.colm.net/files/ragel/ragel-6.10.tar.gz
tar -xzvf ragel-6.10.tar.gz
cd ragel-6.10
./configure
make
sudo make install
```

Hyperscan requires the Boost C++ Libraries. Note that we are not using the Ubuntu repository version of the boost headers (libboost-all-dev) because Hyperscan requires boost libraries at or above version number 1.58, and the Ubuntu repository version is too old. Download the Boost 1.67.0 libraries, but do not install:

```
cd ~/snort_src
wget https://dl.bintray.com/boostorg/release/1.67.0/source/boost_1_67_0.tar.gz
tar -xvzf boost_1_67_0.tar.gz
```

Install Hyperscan 4.7.0 from source, referencing the location of the Boost headers source directory:

```
cd ~/snort_src
wget https://github.com/intel/hyperscan/archive/v4.7.0.tar.gz
tar -xvzf v4.7.0.tar.gz
mkdir ~/snort_src/hyperscan-4.7.0-build
cd hyperscan-4.7.0-build/

cmake -DCMAKE_INSTALL_PREFIX=/usr/local -DBOOST_ROOT=~/snort_src/boost_1_67_0/ ../hyperscan-4.7.0

make
sudo make install
```

If you want to test that Hyperscan works, from the build directory, run:

```
cd ~/snort_src/hyperscan-4.7.0-build/
./bin/unit-hyperscan
```

Snort has an optional requirement for flatbuffers, A memory efficient serialization library:

```
cd ~/snort_src
wget https://github.com/google/flatbuffers/archive/v1.9.0.tar.gz -O flatbuffers-v1.9.0.tar.gz
tar -xzvf flatbuffers-1.9.0.tar.gz
mkdir flatbuffers-build
cd flatbuffers-build

cmake ../flatbuffers-1.9.0

make
sudo make install
```

Next, download and install Data AcQuisition library (DAQ) from the Snort website. Note that Snort 3 uses a different DAQ than the Snort 2.9.x.x series:

```
cd ~/snort_src
wget https://www.snort.org/downloads/snortplus/daq-2.2.2.tar.gz
tar -xvzf daq-2.2.2.tar.gz
cd daq-2.2.2
./configure
make
sudo make install
```

Update shared libraries:

```
sudo ldconfig
```

Clone the latest Snort3 master from github and install:

```
cd ~/snort_src
git clone git://github.com/snortadmin/snort3.git
cd snort3

./configure_cmake.sh --prefix=/usr/local --enable-tcmalloc
cd build
make
sudo make install
```

Note: you may see that Snort does not enable SafeC when you run the make command, this is a bug and can be ignored.

Optional: If you are interested in seeing what additional options can be configured when building Snort, run **./configure_cmake.sh --help** for a full list. Some options you may be interested in is the Snort3 command line shell (**--enable-shell**) or support for pcap files over 2 GB (**--enable-large-pcap**).

NOTE: The **git clone** commands above download the most recent version of Snort++ from github (which is probably what you want). If you instead want to download a specficif build release, check the github repository.

I doubt you'd want to do this, but if you're having issues, this can help identify if the problem is that a newer version of Snort++ is installing differently. Snort++ is still evolving as it is developed, and the dependencies it relies on keep changing, sometimes with each release.

Next, we want to test that Snort runs:

```
noah@snort3:~$ /usr/local/bin/snort -V

   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.0.0 (Build 245) from 2.9.11
   ''''    By Martin Roesch & The Snort Team
           http://snort.org/contact#team
           Copyright (C) 2014-2018 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using DAQ version 2.2.2
           Using LuaJIT version 2.1.0-beta3
           Using OpenSSL 1.1.0g  2 Nov 2017
           Using libpcap version 1.8.1
           Using PCRE version 8.39 2016-06-14
           Using ZLIB version 1.2.11
           Using FlatBuffers 1.9.0
           Using Hyperscan version 4.7.0 2018-06-12
           Using LZMA version 5.2.2
```

Finally it's good practice to create a link to snort in /usr/local/sbin:

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

3 Running Snort

Snort 3 requires a few environmental variables in order to run correctly. We store these variables temporarily in the current session and save them permanently to the our local .bashrc file (note that LUA_PATH can't be stored in /etc/profile because it won't load correctly. You'll need to run these lines for every user who needs to run Snort on this system):

```
export LUA_PATH=/usr/local/include/snort/lua/\?.lua\;\;
export SNORT_LUA_PATH=/usr/local/etc/snort

sh -c "echo 'export LUA_PATH=/usr/local/include/snort/lua/\?.lua\;\;' >> ~/.bashrc"
sh -c "echo 'export SNORT_LUA_PATH=/usr/local/etc/snort' >> ~/.bashrc"
```

To make these environmental variables available when you use sudo, add them to the /etc/sudoers file:

```
sudo visudo
```

add the following line to the end:

```
Defaults env_keep += "LUA_PATH SNORT_LUA_PATH"
```

use ctrl-x to exit, save when prompted by pressing y, then press enter to save the file to /etc/sudoers.tmp (which will get copied automatically to /etc/sudoers).

Now lets test Snort with the default configuration file:

```
snort -c /usr/local/etc/snort/snort.lua
```

You should see output that finishes with the following:

```
Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting
```

4 Install Locations

A note on install locations:

When we ran **./configure --prefix=/usr/local** we were telling snort to install all files under the /usr/local folder. Alternately, you could install Snort to its own folder to keep all files in one place (recommended for testing), such as /opt/snort, to keep all the files installed by Snort under one folder without mixing files from other installs. /usr/local is the proper location to install software compiled from scratch.

Here are a number of the snort files and folders that are created (I've removed non-snort files and folders for clarity)

```
noah@snort3:~$ tree /usr/local/ -L 3
 bin
    daq-modules-config
    fbstreamer
    snort
    snort2lua
    u2boat
    u2spewfoo
 etc
    snort
        file_magic.lua
        snort_defaults.lua
        snort.lua
 include
    daq_api.h
    daq_common.h
    daq.h
    snort
        actions
        codecs
        daqs
          (...)
 lib
    daq
       daq_afpacket.la
         (...)
    odp
       appid.conf
       appMapping.data
    snort
       daqs
    snort_extra
        codecs
        daqs
        loggers
        search_engines
 share
     doc
         snort
```

The bin folder contains the following files:
fbstreamer: A utility for accessing the statistics generated in flatbuffer format.
snort: The Snort binary.
snort2lua: Tool to convert a Snort 2.9.x.x configuration file into a 3.x configuration file.
u2boat: U2boat is a tool for converting unified2 files into different formats.
u2spewfoo: U2SpewFoo is a lightweight tool for dumping the contents of unified2 files to stdout.

Additionally, the following folders are created and used:

/usr/local/bin Binaries for Snort and supporting software.
/usr/local/etc/snort: The configuration files for Snort.
/usr/local/include/snort: All include files for Snort.
/usr/local/lib/pkgconfig: The pkgconfig file for Snort (compilation details for Snort).
/usr/local/share/doc/snort: The documentation for the installed version of Snort.

5 Snort Rules

Snort3 rules have more options than Snort 2 rules, and while the normal rules downloaded with PulledPork or manually will work, for testing you will probably want to download the set of community rules specifically created for snort3. You can manually download snort3 specific community rules from the snort website:

```
cd ~/snort_src/
wget https://www.snort.org/downloads/community/snort3-community-rules.tar.gz
tar -xvzf snort3-community-rules.tar.gz
cd snort3-community-rules

sudo mkdir /usr/local/etc/snort/rules
sudo mkdir /usr/local/etc/snort/builtin_rules

sudo mkdir /usr/local/etc/snort/so_rules
sudo mkdir /usr/local/etc/snort/lists

sudo cp snort3-community.rules /usr/local/etc/snort/rules/
sudo cp sid-msg.map /usr/local/etc/snort/rules/
```

now test that the rules load properly:

```
snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort/rules/snort3-community.rules
```

The output should now show loaded rule counts, a snippet:

```
Loading rules:
Loading /usr/local/etc/snort/rules/snort3-community.rules:
Finished /usr/local/etc/snort/rules/snort3-community.rules.
Finished rules.
--------------------------------------------------
rule counts
rule counts
       total rules loaded: 829
              text rules: 829
            option chains: 829
            chain headers: 46
--------------------------------------------------
...
```

Note: many of the rules in the snort3-community rules are commented out. This is explained here, but basically the reason is that some of the rules could generate false positives, so the Snort team has commented out those rules, since they can flood your logs or lead to excessive traffic interruptions if you're running Snort in NIPS mode. To uncomment these rules, please use the following command:

```
sudo sed -i '17,$s/^# //' /usr/local/etc/snort/rules/snort3-community.rules
```

you may want to run Snort with the following flags to detect issues, the **warn-all** and **pedantic** flags. From the Snort3 manual:

Warnings are not emitted unless --warn- is specified. --warn-all enables all warnings, and --pedantic makes such warnings fatal*

you will not want to use the **--pedantic** flag when running Snort, as simple flowbit warnings (flowbits set but not used in a rule, a common issue) will generate warnings and cause Snort to error out. It is a good flag for testing your configuration though.

6 Snort Builtin Rules

To enable the enable decoder and inspector alerts (these detect on malicious traffic that can't be easily detected with regular rules), we need to enable this option in our snort configuration file: **snort.lua**, located in the **/usr/local/etc/snort/** directory:

To edit your **snort.lua** configuration file, sudo is required):

```
sudo vi /usr/local/etc/snort/snort.lua
```

and at line 195, set **enable_builtin_rules** to true. Lines that start with two hyphens are comments (or commands that are commented out), and are not parsed by snort when loaded. Remove the two hyphens before **enable_builtin_rules** to enable this option. All indented lines in your snort.lua **must** be four spaces (not a tab) or the configuration will not load. Your ips module should look like this (shortened for clarity):

```
ips =
{
    -- use this to enable decoder and inspector alerts
    enable_builtin_rules = true,
}
```

Now test:

```
snort -c /usr/local/etc/snort/snort.lua
```

you should see the builtin rules loaded (we didn't load the community rules, shown as **text** rules above):

```
--------------------------------------------------
rule counts
       total rules loaded: 471
            builtin rules: 471
            option chains: 471
            chain headers: 1
--------------------------------------------------
```

Now let's load both types of rules (the builtin and the community rules):

```
snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort/rules/snort3-community.rules
```

and you'll see both builtin rules and text rules loaded:

```
-------------------------------------------------
rule counts
       total rules loaded: 3959
              text rules: 3488
            builtin rules: 471
            option chains: 3959
            chain headers: 265
-------------------------------------------------
```

7 Installing OpenAppID

OpenAppID allows for the identification of application layer traffic. You can create rules that operate on application-layer traffic (say to block facebook), and to log traffic statistics for each type of traffic detected. The Snort team has put together a package of detectors with assistance from the community that you can download and install, called the Application Detector Package. First download the OpenAppID detector package:

```
cd ~/snort_src/
wget https://www.snort.org/downloads/openappid/7630 -O OpenAppId-7630.tar.gz
tar -xzvf OpenAppId-7630.tar.gz
sudo cp -R odp /usr/local/lib/
```

Now we need to edit our Snort configuration file to point to this odp directory:

```
sudo vi /usr/local/etc/snort/snort.lua
```

At line 114 (yours line number may be slightly different) you will see the **appid =** entry. You will want to add the app_detector_dir option here, pointing to the parent folder of the odf folder. It should look like this:

```
appid =
{
    -- appid requires this to use appids in rules
    app_detector_dir = '/usr/local/lib',
}
```

note that you must have four spaces (not a tab) for the indented line. Now we want to test the configuration file loads correctly:

```
snort -c /usr/local/etc/snort/snort.lua --warn-all
```

you should see output similar to:

```
...
Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting
```

Now to load Snort with the OpenAppID detectors, as well as all rules (we omit the pedantic flag, since the rules will throw flowbit warnings that are non fatal and can be ignored:

```
snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort/rules/snort3-community.rules --warn-all
```

Create a simple rule to test that OpenAppID is working correctly:

```
sudo touch /usr/local/etc/snort/rules/local.rules
sudo vi /usr/local/etc/snort/rules/local.rules
```

with the following content:

```
alert tcp any any -> any any ( msg:"Facebook trafic Seen"; appids:"Facebook";sid:10000001; )
alert icmp any any -> any any (msg:"ICMP Traffic Detected";sid:10000002;)
```

Now test to make sure the rule loads correctly (is correctly formatted):

```
snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort/rules/local.rules --warn-all
```

If you scroll up through the output, you should see these two text rules loaded successfully. Now let's run snort in detection mode on an interface (change eth0 below to match your interface name), printing alerts to the console:

```
sudo snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort/rules/local.rules \
        -i eth0 -A alert_fast -s 65535 -k none
```

the **-k none** flag tells Snort to ignore bad checksums, and the **-s 65535** flag prevents snort from truncating oversized packets. the Stream and Frag decoders will drop packets that have bad checksums, and the packets will not get processed by the OpenAppID detectors. By including these flags, we ensure that a packet with a bad checksum still gets processed.

now from another window on that computer (open a new terminal window or a second ssh session), use wget to connect to facebook:

```
wget facebook.com
```

from the first console window you will see alerts output similar to the following:

```
08/05-19:13:45.451834 [**] [1:10000001:0] "Facebook trafic Seen" [**] [Priority: 0] [AppID: Facebook] {TCP} 157.240.1.35:443 -> 10.0.0.104:33882
08/05-19:13:45.451842 [**] [1:10000001:0] "Facebook trafic Seen" [**] [Priority: 0] [AppID: Facebook] {TCP} 10.0.0.104:33882 -> 157.240.1.35:443
```

use ctrl-c to stop Snort. You can also ping to or from this machine to generate alerts (the second rule in the local.rules file). This is a good rule for testing.

If you want to collect OpenAppID statistics (how much traffic was detected by each detector), you will need to enable it in the snort.lua file, and run Snort with the **-l** flag (log directory).

First create a log directory

```
sudo mkdir /var/log/snort
```

now modify /usr/local/etc/snort/snort.lua to enable the appid detector to log statistics (line 113):

```
appid =
{
    app_detector_dir = '/usr/local/lib',
    log_stats = true,
}
```

Now run snort, listening to an interface, logging data to the /var/log/snort folder

```
sudo snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/snort/rules/local.rules \
        -i eth0 -A alert_fast -s 65535 -k none -l /var/log/snort
```

after collecting data and stopping snort, you will see **appid_stats.log** in /var/log/snort. This file is owned by root, so make it readable by all:

```
sudo chmod a+r /var/log/snort/appid_stats.log
```

now you can look at the protocol statistics that snort collected:

```
noah@snort3:~/snort_src$ cat /var/log/snort/appid_stats.log
1513397914,DNS,296,442
1513397914,Facebook,2065,5356
1513397914,HTTP,543,586
1513397914,Wget,543,586
1513397914,HTTPS,1522,4770
1513397914,SSL client,1522,4770
1513397914,ICMP for IPv6,156,0
```

This is a comma-separated file that shows the time (unixtime), detector, sent bytes (tx), and received bytes (rx), in that order. if you don't want this data collected, you can disable the log_stats option in the appid module in your snort.lua configuration file.

For more information on detectors, please see the OpenAppID detector Guide.

8 Installing Example Plugins

If you want to develop Snort plugins, you will want to compile and install the **snort-extras** package. If you installed Snort 3 to a different location than /usr/local, you'll need to change the PKG_CONFIG_PATH command below to point to the different location.

Navigate to the extras directory, compile, and install:

```
cd ~/snort_src/
git clone https://github.com/snort3/snort3_extra.git
cd ./snort3_extra/
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig/
./configure_cmake.sh --prefix=/usr/local
cd build
make
sudo make install
```

If you want Snort to use the newly added plugins, you need to pass it the **--plugin path** and/or the **--script path** options (plugins can be written in c++ or in lua).

For example, to load the **alert_ex** plugin:

```
snort --plugin-path /usr/local/lib/snort_extra -A alert_ex --warn-all
```

(the warn-all flag is there to catch any errors).

if you want to test a lua script plugin:

```
snort --script-path /usr/local/lib/snort_extra -A lualert --warn-all
```

To see all available plugins (excluding the plugins from the snort-extras package), use the following command:

```
snort --list-plugins
```

Now to have snort list all the new plugins including those it can see in the extras directory:

```
snort --script-path /usr/local/lib/snort_extra --plugin-path /usr/local/lib/snort_extra \
        --list-plugins
```

please see the snort3 extras readme and the included source files for more information.

to see the number of new plugins the extras folder makes available, let's show all the logging modules by default, then with the extras enabled:

```
noah@snort3:~$ snort --list-plugins | grep logger
logger::alert_csv v0 static
logger::alert_fast v0 static
logger::alert_full v0 static
logger::alert_json v0 static
logger::alert_sfsocket v0 static
logger::alert_syslog v0 static
logger::alert_unixsock v0 static
logger::log_codecs v0 static
logger::log_hext v0 static
logger::log_pcap v0 static
logger::unified2 v0 static

noah@snort3:~$ snort --script-path /usr/local/lib/snort\_extra \
        --plugin-path /usr/local/lib/snort\_extra --list-plugins | grep logger
logger::alert_csv v0 static
logger::alert_ex v0 /usr/local/lib/snort_extra/loggers/alert_ex.so
logger::alert_fast v0 static
logger::alert_full v0 static
logger::alert_json v0 static
logger::alert_sfsocket v0 static
logger::alert_syslog v0 static
logger::alert_unixsock v0 static
logger::log_codecs v0 static
logger::log_hext v0 static
logger::log_null v0 /usr/local/lib/snort_extra/loggers/log_null.so
logger::log_pcap v0 static
logger::lualert v0 static
logger::unified2 v0 static
```

9 Snort++ Developers Guide

If you are interested in developing a snort plugin or module, you will want to build the Snort++ Developers Guide. This requires a few software packages (the optional documentation packages from earlier in this guide). Note that these packages are nearly 800 MB in size:

```
sudo apt-get install -y asciidoc dblatex source-highlight w3m
```

now build the guide

```
cd ~/snort_src/snort3
./doc/dev_guide.sh
```

you now have **dev_guide.html** in your current directory (snort_src/snort3)

10 Example PCAP files to generate alerts to CSV file

If you are looking for PCAP files that can be used to generate alerts from your builtin and community rules for testing purposes, I have found that MACCDC 2012 dataset is good. Start by downloading two of the pcap files:

```
cd ~
mkdir pcaps
cd pcaps

wget https://download.netresec.com/pcap/maccdc-2012/maccdc2012_00000.pcap.gz
gunzip maccdc2012_00000.pcap.gz

wget https://download.netresec.com/pcap/maccdc-2012/maccdc2012_00001.pcap.gz
gunzip maccdc2012_00001.pcap.gz
```

Now we run snort, telling it load the first pcap file that we downloaded, load all community rules, and print alerts to the console (I break down all the flags used here if you scroll down). Note that this may take some time to run, you can use ctrl-c to stop if if you're impatient (this took about 4 minutes to run on my system, generating over 187,000 alerts):

```
sudo snort -c /usr/local/etc/snort/snort.lua -r ~/pcaps/maccdc2012_00000.pcap -R \
       /usr/local/etc/snort/rules/snort3-community.rules -A alert_fast -s 65535 -k none
```

To process multiple pcap files at once, modify the last command to scan all pcap files in that same directory with the following command:

```
sudo snort -c /usr/local/etc/snort/snort.lua --pcap-filter \*.pcap --pcap-dir ~/pcaps -R \
       /usr/local/etc/snort/rules/snort3-community.rules -A alert_fast -s 65535 -k none
```

We're doing a lot here, let's break down this command:

sudo snort	This is the snort binary we are calling. Run with sudo since we want to write to /var/log/snort next
-c /usr/local/etc/snort/snort.lua	This is our snort.lua configuration file
--pcap-filter *.pcap	This tells snort how to identify pcap files located in the pcap-dir
--pcap-dir /pcaps	This tells snort where to look for pcap files
-R /usr/local/etc/snort/rules/snort3-community.rules	This is the rule file to load (in addition to the builtin rules)
-s 65535	Set the snaplen so Snort doesn't truncate and drop oversized packets
-k none	Ignore bad checksums, otherwise snort will drop packets with bad checksums, and they won't be evaluated

When I ran this command, snort generated 234,328 alerts (look in the output at the end for Module Statistics: Detection: total_alerts) in six minutes (look in the output for Summary Statistics: timing: seconds).

You can modify this last command to use any output plugin (we used the alert_fast plugin), either by specifying it at the command line, or alternately by enabling the plugin in your **snort.lua** file (line 250). For example, to use the alert_csv plugin to save alert data to a csv file, you would modify your snort.lua to look like this (line 253):

```
alert_csv =
{
    file = true,
}
```

then run snort as follows:

```
sudo snort -c /usr/local/etc/snort/snort.lua -r ~/pcaps/maccdc2012_00000.pcap \
        -R /usr/local/etc/snort/rules/snort3-community.rules -l /var/log/snort -s 65535 -k none -q
```

you'll note that we don't specify the output plugin on the command line, that's because we have enabled the alert_csv plugin in our snort.lua, and specifying one on the command line would take precedence over any output plugins configured in our snort.lua. We are specifying the log directory though, with the **-l** flag.

after running the command, you should see alert_csv.txt in your /var/log/snort directory. You'll want to make this file readable by all:

```
sudo chmod a+r /var/log/snort/alert_csv.txt
```

using the **wc** command, we can see how many alerts were generated from these pcap files (this should be the same as when we output to the console):

```
noah@snort3:~$ wc -l /var/log/snort/alert_csv.txt
187460 /var/log/snort/alert_csv.txt
noah@snort3-external:~$
```

There are more options that this plugin (and other plugins) have. All this information and more is available in the snort3 manual.

11 Testing Snort3 with Bats: Bash Automated Testing System

If you're running Ubuntu 16 or 18 (not 14), you can run the Snort 3 automated tests. These tests use Bats.

```
sudo apt-get install -y bats
cd snort_src/
git clone https://github.com/snort3/snort3_demo.git
cd snort3_demo/
./run_test.sh /usr/local
```

If you're running Ubuntu 14, you could probably install Bats from source if you wanted.

12 Where to Go from Here

See the Snort 3 manual for more information about running Snort 3 and compilation options. Snort3 is much different from the Snort 2.9.9.x series, and reading the manual is highly recommended. Both configuration

and rule files are different, and not compatible between the two versions. Old Snort 2 configuration and rule files can be converted to the Snort3 format using the included snort2lua command.

I have tutorials available on my website for configuring a fully-featured Snort system on Ubuntu, including Barnyard2, PulledPork, and BASE, configuring Snort to run as a NIPS (dropping / blocking malicious traffic), and configuring OpenAppID for layer 7 application detection. These tutorials and more are available at SublimeRobots.com.

If you are looking to load your alert data into a SIEM (Splunk, ELK, or others), the Snort team is developing a JSON plugin that will output alerts to the log directory in JSON format, which is perfect for these types of tool to ingest. The alert_json plugin is part of the Example Plugins package, and a good guide for setting it up can be found in this blog post: Snort 3.0 with ElasticSearch, LogStash, and Kibana (ELK).

Feedback: Please send me feedback with issues you encountered and recommendations for changes to this guide: Noah@SublimeRobots.com. Feedback helps me to update these guides, and helps me identify common issues and questions that people encounter when running through these instructions.