Snort 3 on CentOS 8

Generated: 2019-09-30

Table of Contents

1. Introduction

This guide walks through installing and configuring Snort 3 on CentOS 8. Some of the configurations may not be applicable to production sensors. Therefore, the steps in this guide should be tested first.

This guide was tested on CentOS 8 image:

Base Image	:	`CentOS-8-x86_64-1905-dvd1.iso`
Release	:	`CentOS Linux release 8.0.1905 (Core)`
Kernel	:	`4.18.0-80.7.1.el8_0.x86_64`
Software	:	`Minimal Install`

Snort 3 information:

Build	:	`261`
Source	:	`git`

LibDAQ information:

Build	:	`3.0.0`
Source	:	`git`

The following conventions are used for installing and configuring Snort.

Snort install prefix	`/usr/local/snort`
Rules directory	`/usr/local/snort/rules`
AppID directory	`/usr/local/snort/appid`
IP Reputation lists directory	`/usr/local/snort/intel`
Logging directory	`/var/log/snort`
Snort Extra Plugins directory	`/usr/local/snort/extra`

2. Preparation

With CentOS 8, several development packages and headers (for example, `libpcap-devel`) required for successfully compiling LibDAQ and Snort are not included in the default repositories – `AppStream`, `Base`, or `Extras`. Instead, they exist in the `PowerTools` repository, which is disabled by default.

```
# dnf config-manager --add-repo /etc/yum.repos.d/CentOS-PowerTools.repo
# dnf config-manager --set-enabled PowerTools
```

Another change with CentOS 8 is that some packages were removed from the official repositories, such as gperftools and unwind [1].

Some development packages such as LuaJIT and unwind exist in the EPEL repository. This streamlines the installation and future updates of these packages. If installing EPEL repository is not an option, the associated packages can be installed from source code.

```
# dnf install epel-release
```

Now that all of the repositories enabled, it is time to ensure that the operating system and existing packages are up to date. This may require a reboot, especially if the updates included kernel upgrades.

```
# dnf upgrade
# reboot now
```

Since some of the packages maybe built from source, a directory is created to house the source codes.

```
# mkdir sources && cd sources
```

Next, some helper packages are installed, which are not required by Snort and can be removed later.

```
# dnf install vim git
```

Basic compilation tools required for building Snort are installed from the repository. These include: **flex** (`flex`), **bison** (`bison`), **gcc** (`gcc`), **c++** (`gcc-c++`), **make** (`make`), and **cmake** (`cmake`). Unlike CentOS 7, installing cmake from source code is not required since the cmake version in CentOS 8 is compatible. Additionally, **autoconf** (`autoconf`) and **libtool** (`libtool`) packages will be installed in order to successfully compile LibDAQ.

```
# dnf install flex bison gcc gcc-c++ make cmake autoconf libtool
```

3. Installing Snort 3 Dependencies

CentOS 8 improves upon packages versions included in the repositories. As a result, these packages can be installed directly from the repository as opposed to compiling them from source in CentOS 7. The following table summarizes the required and optional packages for compiling Snort and LibDAQ.

Dependency	Status	Source	Dependency	Status	Source
`dnet`	Required	Repository	`LibDAQ`	Required	Source Code
`pcap`	Required	Repository	`lzma`	Optional	Repository
`pcre`	Required	Repository	`hyperscan`	Optional	Source Code
`OpenSSL`	Required	Repository	`flatbuffers`	Optional	Source Code
`zlib`	Required	Repository	`safec`	Optional	Source Code
`pkgconfig`	Required	Repository	`uuid`	Optional	Repository
`LuaJIT`	Required	Repository	`tcmalloc`	Optional	Source Code
`hwloc`	Required	Repository	`libmnl`	Optional	Repository

Other packages and their purposes can be viewed at [2].

3.1 Required Dependencies

The following packages are installed from CentOS repositories: **pcap** (`libpcap-devel`), **pcre** (`pcre-devel`), **dnet** (`libdnet-devel`), **hwloc** (`hwloc-devel`), **OpenSSL** (`openssl-devel`), **pkgconfig** (`pkgconfig`), **zlib** (`zlib-devel`), and **LuaJIT** (`luajit-devel`).

```
# dnf install libpcap-devel pcre-devel libdnet-devel hwloc-devel openssl-devel zlib-devel luajit-
devel pkgconfig
```

LibDAQ

Recent revisions of Snort 3 require the new LibDAQ (>=3.0.0). If building LibDAQ with NFQ module support, then the following packages must be installed before configuration: **libnfnetlink** (`libnfnetlink-devel`), **libnetfilter_queue** (`libnetfilter_queue-devel`), and **libmnl** (`libmnl-devel`).

```
# dnf install libnfnetlink-devel libnetfilter_queue-devel libmnl-devel
```

Clone LibDAQ from GitHub and generate the configuration script – since it is cloned from git.

```
# git clone https://github.com/snort3/libdaq.git
# cd libdaq/
# ./bootstrap
```

Otherwise, proceed to the configuration steps taking into account which modules to disable if not used. Now it's time to configure LibDAQ, which should result in a similar output (omitted) as shown below.

```
# ./configure
```

```
...
Build AFPacket DAQ module.. : yes
Build BPF DAQ module....... : yes
Build Divert DAQ module.... : no
Build Dump DAQ module...... : yes
Build FST DAQ module....... : yes
Build NFQ DAQ module....... : yes
Build PCAP DAQ module...... : yes
Build netmap DAQ module.... : no
Build Trace DAQ module..... : yes
```

Proceed to installing LibDAQ.

```
# make
# make install
# cd ../
```

3.2 Optional Dependencies

LZMA and UUID

`lzma` is used for decompression of SWF and PDF files, while `uuid` is a library for generating/parsing Universally Unique IDs for tagging/identifying objects across a network.

```
# dnf install xz-devel libuuid-devel
```

Hyperscan

Prior to installing hyperscan, the following dependencies should be installed: **Ragel**, **Boost,** and **sqlite3** (`sqlite-devel`). CentOS 8 does not come with Python preinstalled. Building hyperscan requires a python interpreter, **python3** (`python3`) available on the host. Both, python3 and sqlite will be installed from the repository.

```
# dnf install python3 sqlite-devel
```

Installing newer versions (>=7.x) of Ragel requires installing colm first. Prior versions, for example version 6.10, do not require installing colm. The steps will proceed with installing colm (0.13.0.7) and ragel (7.0.0.12).

```
# curl -LO http://www.colm.net/files/colm/colm-0.13.0.7.tar.gz
# tar xf colm-0.13.0.7.tar.gz && cd colm-0.13.0.7
# ./configure
# make -j$(nproc)
# make -j$(nproc) install
# ldconfig
# cd ../
```

```
# curl -LO http://www.colm.net/files/ragel/ragel-7.0.0.12.tar.gz
# tar xf ragel-7.0.0.12.tar.gz && cd ragel-7.0.0.12
# ./configure
# make -j$(nproc)
# make -j$(nproc) install
# ldconfig
# cd ../
```

The remaining dependency is boost, which will be downloaded and decompressed without building it.

```
# curl -LO https://dl.bintray.com/boostorg/release/1.71.0/source/boost_1_71_0.tar.gz
# tar xf boost_1_71_0.tar.gz
```

Download and install Hyperscan (5.2.0):

```
# curl -Lo hyperscan-5.2.0.tar.gz https://github.com/intel/hyperscan/archive/v5.2.0.tar.gz
# tar xf hyperscan-5.2.0.tar.gz
# mkdir hs-build && cd hs-build
```

There are two methods to make hyperscan aware of the Boost headers: 1) `Symlink`, **or** 2) Passing `BOOST_ROOT` pointing to the root directory of the Boost headers to cmake. Both methods are shown below.

Method 1 – `Symlink`:

```
# ln -s ~/sources/boost_1_71_0 /boost ~/sources/hyperscan-5.2.0/include/boost
# cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local ../hyperscan-5.2.0
```

Method 2 – `BOOST_ROOT`:

```
# cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local -DBOOST_ROOT=../boost_1_71_0
../hyperscan-5.2.0
```

Proceed with installing Hyperscan.

```
# make -j$(nproc)
# make -j$(nproc) install
# cd ../
```

Flatbuffers

Flatbuffers is a cross-platform serialization library for memory-constrained apps. It allows direct access of serialized data without unpacking/parsing it first.

```
# curl -Lo flatbuffers-1.11.tar.gz https://github.com/google/flatbuffers/archive/v1.11.0.tar.gz
# tar xf flatbuffers-1.11.tar.gz
# mkdir fb-build && cd fb-build
# cmake ../flatbuffers-1.11.0
# make -j$(nproc)
# make -j$(nproc) install
# cd ../
```

Safec

Safec is used for runtime bounds checks on certain legacy C-library calls.

```
# curl -LO https://github.com/rurban/safeclib/releases/download/v04062019/libsafec-04062019.0-
ga99a05.tar.gz
# tar xf libsafec-04062019.0-ga99a05.tar.gz && cd libsafec-04062019.0-ga99a05
# ./configure
# make
# make install
# cd ../
```

Tcmalloc

tcmalloc is a library created by Google (PerfTools) for improving memory handling in threaded programs. The use of the library may lead to performance improvements and memory usage reduction**.** Neither CentOS 8 nor EPEL repositories include the **gperftools** (`gperftools-devel`) package, therefore, `gperftools` will be built from sources.

Building gperftools from source requires **unwind** (`libunwind-devel`) package to be installed.

```
# dnf install libunwind-devel
```

Proceed to building gperftools.

```
# curl -LO https://github.com/gperftools/gperftools/releases/download/gperftools-2.7/gperftools-
2.7.tar.gz
# tar xf gperftools-2.7.tar.gz && cd gperftools-2.7
# ./configure
# make -j$(nproc)
# make -j$(nproc) install
# cd ../
```

4. Installing Snort 3

Now that all of the dependencies are installed, clone Snort 3 repository from GitHub.

```
# git clone https://github.com/snort3/snort3.git
# cd snort3
```

Before running the configuration step, export the `PKG_CONFIG_PATH` to include the LibDAQ pkgconfig path, as well as other packages' pkgconfig paths, otherwise, the build process may fail or Snort 3 will not be able to locate the associated libraries at compile or runtime.

```
# export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH
# export PKG_CONFIG_PATH=/usr/local/lib64/pkgconfig:$PKG_CONFIG_PATH
```

Note: If LibDAQ or other packages were installed to a custom, non-system path, then that path should be exported to `PKG_CONFIG_PATH`, for example:

```
# export PKG_CONFIG_PATH=/opt/libdaq/lib/pkgconfig:$PKG_CONFIG_PATH
```

Proceed to building Snort 3 while enabling tcmalloc support.

```
# ./configure_cmake.sh --prefix=/usr/local/snort --enable-tcmalloc
```

The above command should result in an output (omitted) similar to below.

```
-------------------------------------------------------
snort version 3.0.0

Install options:
    prefix:     /usr/local/snort
    includes:   /usr/local/snort/include/snort
    plugins:    /usr/local/snort/lib64/snort
...
Feature options:
    DAQ Modules:    Static (afpacket;bpf;dump;fst;nfq;pcap;trace)
    Flatbuffers:    ON
    Hyperscan:      ON
    ICONV:          ON
    LZMA:           ON
    RPC DB:         Built-in
    SafeC:          ON
    TCMalloc:       ON
    UUID:           ON
-------------------------------------------------------
```

Proceed to installing Snort 3.

```
# cd build/
# make -j$(nproc)
# make -j$(nproc) install
# cd ../../
```

Once the installation is complete, verify that Snort 3 reports the expected version and library names

```
# /usr/local/snort/bin/snort -V
```

```
   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.0.0 (Build 261)
   ''''    By Martin Roesch & The Snort Team
           http://snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using DAQ version 3.0.0
           Using LuaJIT version 2.1.0-beta3
           Using OpenSSL 1.1.1 FIPS  11 Sep 2018
           Using libpcap version 1.9.0-PRE-GIT (with TPACKET_V3)
           Using PCRE version 8.42 2018-03-20
           Using ZLIB version 1.2.11
           Using FlatBuffers 1.11.0
           Using Hyperscan version 5.2.0 2019-09-26
           Using LZMA version 5.2.4
```

5. Installing Snort 3 Extras for Additional Capabilities

Snort 3 Extras is a set of C++ or Lua plugins to extend the functionality of Snort 3 in terms network traffic decoding, inspection, actions, and logging. One particular plugin is emphasized and configured in this guide is the `data_log` inspector plugin. The emphasis of this inspector is detailed in a later section.

To install Snort extras, clone its repository from GitHub.

```
# git clone https://github.com/snort3/snort3_extra.git
```

Before building the extra plugins, the environment variable `PKG_CONFIG_PATH` must be set. The path can be verified by listing Snort installation directory.

```
# cd snort3_extra
# export PKG_CONFIG_PATH=/usr/local/snort/lib64/pkgconfig:$PKG_CONFIG_PATH
# ./configure_cmake.sh --prefix=/usr/local/snort/extra
# cd build/
# make -j$(nproc)
# make -j$(nproc) install
# cd ../../
```

6. Configuring Snort 3

Snort 3 includes two main configuration files, `snort_defaults.lua` and `snort.lua`. The file `snort_defaults.lua` contains default values for rules paths, networks, ports, wizards, and inspectors, etc.

The file `snort.lua` is the main configuration file of Snort, allowing the implementation and configuration of Snort inspectors (preprocessors), rules files inclusion, event filters, output, etc. The file `snort.lua` uses the file `snort_defaults.lua` to import defaults values for various Snort configurations.

An additional file `file_magic.lua` exists in the `etc/snort/` directory. This file contains pre-defined file identities based on the hexadecimal representation of the files magic headers. These help Snort identify the file types traversing the network when applicable. This file is also used by Snort main configuration file `snort.lua` and does not require any modifications. The configuration changes and the respective Snort 3 .lua files are shown below.

- Configure rules, reputation, and AppID paths > `snort_defaults.lua`
- Configure HOME_NET and EXTERNAL_NET > `snort.lua`
- Configure ips module > `snort.lua`
- Enable and configure reputation inspector > `snort.lua`
- Configure AppID inspector > `snort.lua`
- Configure file_id and file_log inspectors > `snort.lua`
- Configure data_log inspector > `snort.lua`
- Configure logging > `snort.lua`

Note that Snort inspectors and modules allow variety of customizations and configurations. The configurations made in this section are minimal with the purpose of getting started with Snort 3.

6.1 Global Paths for Rules, AppID, and IP Reputation Lists

Snort rules, appid, and reputation lists will be stored in their respective directory. The `rules/` directory will contain Snort rules files, the `appid/` directory will contain the AppID detectors, and the `intel/` directory will contain IP blacklists and whitelists.

```
# mkdir -p /usr/local/snort/{builtin_rules,rules,appid,intel}
```

Snort Rules

Snort rules consist of text-based rules, and Shared Object (SO) rules and their associated text-based stubs. At the time of writing this guide, the Shared Object rules are not available yet [3].

The rules tarball also contains Snort configuration files. The configuration files from the rules tarball will be copied to the `etc/snort/` directory, and will be used in favor of the configuration files in from Snort 3 source tarball.

To proceed with the configurations, download the rules tarball from Snort.org (PulledPork is not tested yet), replacing the oinkcode placeholder in the below command with the official and dedicated oinkcode.

```
# curl -Lo snortrules-snapshot-3000.tar.gz https://www.snort.org/rules/snortrules-snapshot-
3000.tar.gz?oinkcode=<YOUR OINKCODE HERE>
```

Extract the rules tarball and copy the rules to the `rules/` directory created earlier.

```
# tar xf snortrules-snapshot-3000.tar.gz
# cp rules/*.rules /usr/local/snort/rules/
# cp builtins/builtins.rules /usr/local/snort/builtin_rules/
```

Copy Snort configuration files from the extracted rules tarball `/etc` directory to Snort `etc/snort/` directory.

```
# cp etc/snort_defaults.lua etc/snort.lua /usr/local/snort/etc/snort/
```

OpenAppID (Optional)

Download and extract the OpenAppID package, and move the extracted `odp/` directory to the `appid/` directory.

```
# curl -Lo snort-openappid-11581.tar.gz https://www.snort.org/downloads/openappid/11581
# tar xf snort-openappid-11581.tar.gz
# mv odp/ /usr/local/snort/appid/
```

IP Reputation (Optional)

Download the IP Blacklist generated by Talos and move it to the `intel/` directory created earlier. Enabling the Reputation inspector while in IDS mode will generate blacklist hit alert when a match occurs, and traffic may not be inspected further.

```
# curl -LO https://www.talosintelligence.com/documents/ip-blacklist
# mv ip-blacklist /usr/local/snort/intel/
```

Create an empty file for the IP whitelist, which will be configured along with the blacklist in the following section.

```
# touch /usr/local/snort/intel/ip-whitelist
```

Edit the `snort_defaults.lua` file. The below snapshots of the configurations show the before and after states of the configuration. The paths shown below follow the conventions mentioned at the beginning of this guide.

Change from:

```
---------------------------------------------------------------------------
-- default paths
---------------------------------------------------------------------------
-- Path to your rules files (this can be a relative path)
RULE_PATH = '../rules'
BUILTIN_RULE_PATH = '../builtin_rules'
PLUGIN_RULE_PATH = '../so_rules'

-- If you are using reputation preprocessor set these
WHITE_LIST_PATH = '../lists'
BLACK_LIST_PATH = '../lists'
```

Change to:

```
---------------------------------------------------------------------------
-- default paths
---------------------------------------------------------------------------
-- Path to your rules files (this can be a relative path)
RULE_PATH = '../../rules'
BUILTIN_RULE_PATH = '../builtin_rules'
PLUGIN_RULE_PATH = '../so_rules'

-- If you are using reputation preprocessor set these
WHITE_LIST_PATH = '../../intel'
BLACK_LIST_PATH = '../../intel'

-- Optional
APPID_PATH = '/usr/local/snort/appid'
```

All of the remaining changes will be made in Snort configuration file `snort.lua`

6.2 Configuring HOME_NET and EXTERNAL_NET

The concept of home and external networks in Snort 3 is the same as in Snort 2.X. The changes made below are just an example to demonstrate the syntax.

Change from:

```
-- setup the network addresses you are protecting
HOME_NET = 'any'
```

Change to:

```
-- setup the network addresses you are protecting
HOME_NET = [[ 10.0.0.0/8 192.168.0.0/16 172.16.0.0/12 ]]
```

6.3 Configuring ips Module

The inclusion of Snort rules files (.rules) occurs within the ips module. Using the `snort.lua` copied from the Snort rules tarball, the inclusion of the rules is already configured. As a result, the changes to the ips module are minimal and involves enabling decoder and inspector alerts with the option `--enable_built_rules`, and explicitly defining the ips policy to tap mode. The ips policy governs Snort's operational mode (tap, inline, and inline-test).

Change from:

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

    -- use include for rules files; be sure to set your path
    -- note that rules files can include other rules files
    --include = 'snort3_community.rules'

    -- The following include syntax is only valid for BUILD_243 (13-FEB-2018) and later
    -- RULE_PATH is typically set in snort_defaults.lua
    rules = [[
        include $RULE_PATH/snort3-app-detect.rules
        include $RULE_PATH/snort3-browser-chrome.rules
        .....
        include $RULE_PATH/snort3-sql.rules
        include $RULE_PATH/snort3-x11.rules
    ]]
}
```

Change to:

```
ips =
{
    mode = tap,

    -- use this to enable decoder and inspector alerts
    enable_builtin_rules = true,

    -- use include for rules files; be sure to set your path
    -- note that rules files can include other rules files
    --include = 'snort3_community.rules'

    -- The following include syntax is only valid for BUILD_243 (13-FEB-2018) and later
    -- RULE_PATH is typically set in snort_defaults.lua
    rules = [[
        include $RULE_PATH/snort3-app-detect.rules
        include $RULE_PATH/snort3-browser-chrome.rules
        .....
        include $RULE_PATH/snort3-sql.rules
        include $RULE_PATH/snort3-x11.rules
    ]]
}
```

6.4 Configuring reputation Inspector (Optional)

The reputation inspector is disabled (commented) by default. Uncomment its section and change the values of the `--blacklist` and `--whitelist` variables to point to the paths IP address lists.

Change from:

```
--[[
reputation =
{
    -- configure one or both of these, then uncomment reputation
    --blacklist = 'blacklist file name with ip lists'
    --whitelist = 'whitelist file name with ip lists'
}
--]]
```

Change to:

```
reputation =
{
    -- configure one or both of these, then uncomment reputation
    blacklist = BLACK_LIST_PATH .. '/ip-blacklist',
    whitelist = WHITE_LIST_PATH .. '/ip-whitelist'
}
```

6.5 Configuring appid Inspector (Optional)

The AppID inspector is enabled by default, however, the path to the AppID package and detector are commented. Uncomment the `app_detector_dir` and change its value the global AppID path defined in the earlier in the `snort_default.lua` file.

Change from:

```
appid =
{
    -- appid requires this to use appids in rules
    --app_detector_dir = 'directory to load appid detectors from'
}
```

Change to:

```
appid =
{
    -- appid requires this to use appids in rules
    app_detector_dir = APPID_PATH,
    log_stats = true
}
```

6.6 Configuring file_id and file_log Inspectors (Optional)

The `file_id` inspector (file_inspect in Snort 2.x) is enabled by default in `snort.lua` with the following configuration options.

```
file_id = { file_rules = file_magic }
```

This allows Snort to identify the type of a file traversing a network stream via the file magic headers. The `file_id` inspector supports HTTP, SMTP, IMAP, POP3, FTP, and SMB protocols. Taking advantage of the `file_id` inspector involves:

- Including the file magic rules. This step is completed in the default form of the inspector.
- Configuring the inspector and define the policy.
- Enabling the inspector logging to generate file events.

The default configuration of the `file_id` inspector is expanded as follows:

```
file_id =
{
    file_rules = file_magic,
    file_policy =
    {
      { when = { file_type_id = 22 }, use = { verdict = 'log', enable_file_signature = true } },
      { when = { sha256 = "E65ECCC561CACE1860638CD0BC745E59058F16349F7455E215BDDF3233355007" }, use = { verdict = 'log' } }
    }
}
```

The above configuration includes the file magic as required in the first step. The file policy is configured to identify files of type PDF via the magic headers in file_magic.lua located in the Snort `etc/snort/` directory.

```
{ type = "PDF", id = 22, category = "PDF files", rev = 1,
  magic = { { content = "| 25 50 44 46|",offset = 0 } } },
```

This means that when the inspector detects a PDF file over a supported protocol, it will generate an event. The file policy is also configured to generate an event when a file with the specified SHA256 traverses the network.

The final step is to enable event logging for the inspector. This is accomplished with the `file_log` inspector. This inspector has two Boolean options that allow logging of packet and system time of file events.

```
file_log =
{
    log_pkt_time = true,
    log_sys_time = false
}
```

6.7 Configuring data_log Inspector (Optional)

The `data_log` plugin is available via the extra plugins installed in an earlier step. The `data_log` is a passive inspector plugin that does not alter data flowing through Snort, instead, it allows for logging additional network data it is subscribed to within Snort 3 processing workflow.

The inspector can be used to log HTTP request or response headers. Recall in Snort 2.X this was possible using the `log_uri` and `log_hostname` configuration options of the `http_inspect` preprocessor. These two options are no long part of Snort 3 http_inspect inspector, and the `data_log` inspector allows for capturing additional data. The captured data is stored into the log file `data.log` within Snort's configured logging directory.

In order to enable the `data_log` inspector, it must be defined in `snort.lua`. The below example will log both HTTP request headers into the `data_log` file and limit the size of the log file to 100MB before a new log file is generated.

```
data_log =
{
    key = 'http_request_header_event',
    limit = 100
}
```

6.8 Configuring logger Module (Optional)

There are various logger modules available in Snort 3 either natively or via the extra plugins. Loggers are disabled (commented) by default. For this guide, the `alert_fast` logger will be used. Enabling this logger is accomplished by uncommenting its section and configuring it to allow logging to a file. By default Snort uses `/var/log/snort` for saving log files. This can also be specified at run time using the `-l` flag.

Change from:

```
--alert_fast = { }
```

Change to:

```
alert_fast =

{
    file = true
}
```

After the configuration is completed, create the log directory for Snort as mentioned earlier.

```
# mkdir -p /var/log/snort
```

7. Running and Testing Snort 3

Running Snort requires setting two environment variables, `LUA_PATH` and `SNORT_LUA_PATH`. These variables point to the lua and configuration directories within the Snort installation prefix.

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

7.1 Running against PCAP Files

A packet capture was generated to help test the customized configurations. The capture contains network traffic consisting of transferring a PDF file over SMTP and HTTP, transferring a binary file of the SHA256 specified earlier in the file policy over HTTP, and ICMP traffic to a test IP address (10.8.8.8) that is manually added to the blacklist. This will allow testing the various configurations made to Snort thus far.

Snort is run against the packet capture via `-r` flag, while specifying the configuration file via `-c` flag, the log directory via `-l` flag, and the extra plugins directory (for the `data_log` inspector) via `--plugin-path`.

```
# /usr/local/snort/bin/snort -c /usr/local/snort/etc/snort/snort.lua -r test.pcap -l
/var/log/snort --plugin-path /usr/local/snort/extra -k none
```

To run Snort against a set of PCAP files stored in a specific directory

```
# /usr/local/snort/bin/snort -c /usr/local/snort/etc/snort/snort.lua --pcap-dir pcaps/ --pcap-
filter '*.pcap' -l /var/log/snort --plugin-path /usr/local/snort/extra -k none
```

The output generated by Snort displays loaded modules, inspectors, status of parsing reputation lists, and rules and their counts.

```
--------------------------------------------------
o")~   Snort++ 3.0.0-261
--------------------------------------------------
Loading /usr/local/snort/etc/snort/snort.lua:
          ssh
          .....
    Processing blacklist file /usr/local/snort/etc/snort/../../intel/ip-blacklist
    Reputation entries loaded: 1467, invalid: 0, re-defined: 0 (from file /usr/local/snort/etc/snort/../../intel/ip-blacklist)
    Processing whitelist file /usr/local/snort/etc/snort/../../intel/ip-whitelist
    Reputation entries loaded: 0, invalid: 0, re-defined: 0 (from file /usr/local/snort/etc/snort/../../intel/ip-whitelist)
          .....
Finished /usr/local/snort/etc/snort/snort.lua.
Loading builtin:
Finished builtin.
Loading rules:
Loading /usr/local/snort/etc/snort/../../rules/snort3-app-detect.rules:
.....
Finished rules.
--------------------------------------------------
rule counts
       total rules loaded: 12760
               text rules: 12267
            builtin rules: 493
            option chains: 12760
            chain headers: 445
--------------------------------------------------
```

After processing the packet capture, Snort displays modules and inspectors counts. Relevant to this guide are the `appid`, `data_log`, `reputation`, and `file_id` inspector statistics. Note that the appid statistics does not report any icmp flows because the reputation inspector blacklisted the icmp flow destined to the test IP address (10.8.8.8) and the icmp flow was not passed through the remaining inspectors for further processing.

With reputation blacklist

```
appid

                 packets: 2869
       processed_packets: 2866
         ignored_packets: 3
          total_sessions: 3
            appid_unknown: 1
--------------------------------------------------
Appid dynamic stats:

firefox: flows: 0, clients: 2, users: 0, payloads 0, misc:
0, incompatible: 0, failed: 0
http: flows: 2, clients: 0, users: 0, payloads 0, misc: 0,
incompatible: 0, failed: 0
smtp: flows: 1, clients: 0, users: 0, payloads 0, misc: 0,
incompatible: 0, failed: 0
thunderbird: flows: 0, clients: 1, users: 0, payloads 0,
misc: 0, incompatible: 0, failed: 0


--------------------------------------------------
data_log
                 packets: 2
--------------------------------------------------
reputation
                 packets: 7
             blacklisted: 1
--------------------------------------------------
File Statistics
--------------------------------------------------
file type stats (files)
         Type               Download    Upload
       MSEXE( 21)           1           0
         PDF( 22)           1           1
            Total           2           1
--------------------------------------------------
file type stats (bytes)
         Type               Download    Upload
       MSEXE( 21)           1123608     0
         PDF( 22)           232533      232533
            Total           1356141     232533
--------------------------------------------------
file signature stats
         Type               Download    Upload
       MSEXE( 21)           1           0
         PDF( 22)           1           1
            Total           2           1
--------------------------------------------------
```

Without reputation blacklist

```
appid

                 packets: 2875
       processed_packets: 2872
         ignored_packets: 3
          total_sessions: 4
            appid_unknown: 2
--------------------------------------------------
Appid dynamic stats:

firefox: flows: 0, clients: 2, users: 0, payloads 0, misc:
0, incompatible: 0, failed: 0
http: flows: 2, clients: 0, users: 0, payloads 0, misc: 0,
incompatible: 0, failed: 0
smtp: flows: 1, clients: 0, users: 0, payloads 0, misc: 0,
incompatible: 0, failed: 0
thunderbird: flows: 0, clients: 1, users: 0, payloads 0,
misc: 0, incompatible: 0, failed: 0
icmp: flows: 1, clients: 0, users: 0, payloads 0, misc: 0,
incompatible: 0, failed: 0
--------------------------------------------------
data_log
                 packets: 2
--------------------------------------------------
reputation
                 packets: 7

--------------------------------------------------
File Statistics
--------------------------------------------------
file type stats (files)
         Type               Download    Upload
       MSEXE( 21)           1           0
         PDF( 22)           1           1
            Total           2           1
--------------------------------------------------
file type stats (bytes)
         Type               Download    Upload
       MSEXE( 21)           1123608     0
         PDF( 22)           232533      232533
            Total           1356141     232533
--------------------------------------------------
file signature stats
         Type               Download    Upload
       MSEXE( 21)           1           0
         PDF( 22)           1           1
            Total           2           1
--------------------------------------------------
```

Snort also created four different log files in the specified log directory. These logs include events generated by the `ips` module, `appid`, `data_log` and `file_id` inspectors.

```
# ls -l /var/log/snort/
```

```
-rw-------. 1 root root 965 Mar 14 18:41 alert_fast.txt
-rw-------. 1 root root 128 Mar 14 18:41 appid_stats.log
-rw-------. 1 root root 349 Mar 14 18:41 data_log
-rw-------. 1 root root 617 Mar 14 18:41 file.log
```

In this test, the `alert_fast.txt` log file contains events generated by the built-in rules via the `http_inspect` and `reputation` inspectors. The test packet capture did not contain traffic that would trigger event from the text-based rules. The `reputation` inspector generated an alert against a test IP address (10.8.8.8) that was added to the blacklist file.

```
# cat /var/log/snort/alert_fast.txt
```

```
10/13-16:55:43.741000 [**] [119:18:1] "(http_inspect) webroot directory traversal" [**] [Priority: 3] {TCP} 192.168.0.1:14685 -> 173.37.145.84:80
03/07-21:01:28.167818 [force_block] [**] [136:1:1] "(reputation) packets blacklisted" [**] [Priority: 3] {ICMP} 172.24.1.78 -> 10.8.8.8
```

The `appid_stats.log` contains detected apps and protocols and associated statistics. Snort was able to detect the use of Firefox and Thunderbird apps and protocols HTTP and SMTP used to transfer the files.

```
# cat /var/log/snort/appid_stats.log
```

With reputation blacklist	Without reputation blacklist
`1520445690,Firefox,62622,1418464`	`1520445690,Firefox,62514,1418464`
`1520445690,HTTP,62622,1418464`	`1520445690,HTTP,62514,1418464`
`1520445690,SMTP,335494,16292`	`1520445690,SMTP,335440,16292`
`1520445690,Thunderbird,335494,16292`	`1520445690,Thunderbird,335440,16292`
	`1520445690,ICMP,294,294`

The `file.log` file contains events generated by the `file_`id inspector. The events match the configured `file_policy` to detect/log PDF files and the SHA256 hash of one of the files. The first 2 events detect PDF files over SMTP and HTTP respectively. The last event is generated by detecting the specified SHA256 over HTTP.

```
# cat /var/log/snort/file.log
```

```
03/07-21:59:35.125362 10/13-16:55:36.793000  192.168.0.1:14685 -> 173.37.145.84:25, [Name:
"../../file_2_pcap_snort3/file_1.pdf"] [Verdict: Log] [Type: PDF] ❶

03/07-21:59:35.260333 10/13-16:55:44.143000  192.168.0.1:14685 -> 173.37.145.84:80, [Name:
"/file2pcap/%2e%2e%2f%2e%2e%2ffile_2_pcap_snort3%2ffile_1%2epdf"] [Verdict: Log] [Type: PDF] ❷

03/07-21:59:35.465802 10/13-16:56:00.741000  192.168.0.1:9208 -> 173.37.145.84:80, [Name:
"/file2pcap/%2e%2e%2f%2e%2e%2ffile_2_pcap_snort3%2ffile_2%2eexe"] [Verdict: Log] [Type: MSEXE] [SHA:
E65ECCC561CACE1860638CD0BC745E59058F16349F7455E215BDDF3233355007] [Size: 1123608] ❸
```

The `data_log` file contains the HTTP request header logged by the `data_log` inspector. The log file contains 2 log lines since the test packet capture contained only 2 HTTP transaction. The fields are comma-separated and consist of request timestamp, source and destination IPs and ports, host, request URI, and the client user-agent.

```
Mon Oct 13 13:55:36 2008, 192.168.0.1, 9208, 173.37.145.84, 80, wrl, /../file_2_pcap_snort3/file_2.exe,
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.17) Gecko/20081007 Firefox/2.0.0.17

Mon Oct 13 13:55:43 2008, 192.168.0.1, 14685, 173.37.145.84, 80, wrl, /../file_2_pcap_snort3/file_1.pdf,
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.17) Gecko/20081007 Firefox/2.0.0.17
```

Reconfiguring the `data_log` inspector to log the HTTP response headers generates the following log lines. The fields include the request timestamp, source and destination IPs and ports, server header from the response, request URI, and HTTP status code.

```
Mon Oct 13 13:55:36 2008, 192.168.0.1, 9208, 173.37.145.84, 80, Apache/2.2.3 (Debian) PHP/5.2.0-8+etch10
mod_ssl/2.2.3 OpenSSL/0.9.8c, /../file_2_pcap_snort3/file_2.exe, 200

Mon Oct 13 13:55:43 2008, 192.168.0.1, 14685, 173.37.145.84, 80, Apache/2.2.3 (Debian) PHP/5.2.0-8+etch10
mod_ssl/2.2.3 OpenSSL/0.9.8c, /../file_2_pcap_snort3/file_1.pdf, 200
```

7.2 Running against an Interface

Snort can be run against a listening interface via the `-i` flag while specifying the capture network interface.

```
# /usr/local/snort/bin/snort -c /usr/local/snort/etc/snort/snort.lua -i eth0 -l /var/log/snort --
plugin-path /usr/local/snort/extra -k none
```

7.3 Running Snort 3 Demo

Snort 3 demo contains usage examples and tests against Snort 3 in an automated fashion using bats – Bash Automated Testing System. Bats can be installed using the below steps.

```
# git clone https://github.com/sstephenson/bats.git
# cd bats/
# ./install.sh /usr/local
```

Now, clone Snort 3 demo project and run the tests.

```
# git clone https://github.com/snort3/snort3_demo.git
# cd snort3_demo/
# ./run_test.sh /usr/local/snort
```

8. References

[1] https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/considerations_in_adopting_rhel_8/index

[2] https://github.com/snort3/snort3/blob/master/doc/tutorial.txt

[3] http://blog.snort.org/2018/02/snort-30-ruleset-announcement.html

[4] https://www.snort.org/downloads/snortplus/snort_manual.html

[5] https://github.com/snortadmin/snort3/tree/master/doc