# Snort 3 on CentOS 7

**Generated:** 2018-08-29

This guide walks through installing and configuring Snort 3 on CentOS 7. Some of the configured options may not be applicable to all production sensors. Therefore, the steps in this guide should be implemented in a test environment first.

This guide was tested on CentOS 7 image:

| | |
|---|---|
| Base Image | : `CentOS-7-x86_64-Minimal-1804.iso` |
| Release | : `CentOS Linux release 7.5.1804 (Core)` |
| Kernel | : `3.10.0-862.11.6.el7.x86_64` |

Snort 3 information:

| | |
|---|---|
| Build | : `247 (Beta)` |
| Source | : `git clone` |

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` |

This guide is broken into the following sections:

1. **Preparation:** this sections discusses setting up the basic requirements on the test host in order to compile and install Snort 3
2. **Installing Snort 3 Dependencies:** this section is broken into two subsections discussing the required and optional Snort 3 dependencies.

   2.1 Required Dependencies
   2.2 Optional Dependencies

3. **Installing and Verifying Snort 3 Installation:** this is the section in which Snort 3 is installed and its installation is verified.
4. **Installing Snort 3 Extra Plugins for Additional Capabilities:** this section discusses installing Snort 3 extra plugins and the additional functionality they provide to Snort 3 in a Snort 3 deployment scenario.
5. **Configuring Snort 3:** this section looks at configuring select modules and inspectors of Snort 3. Some of these configurations may not be apply to all deployment scenarios. This section is further broken into the following subsections.

   5.1 Global Paths for Rules, AppID, and IP Reputation
   5.2 Setting up HOME_NET and EXTERNAL_NET
   5.3 ips Module
   5.4 reputation Inspector
   5.5 appid Inspector
   5.6 file_id and file_log Inspectors
   5.7 data_log Inspector
   5.8 logger Module

6. **Running and Testing Snort 3:** this section is dedicated to testing Snort 3 installation and the configurations made in previous sections.

   6.1 Running against a PCAP
   6.2 Running against an Interface

7. **References**

# 1. Preparation

Ensure that the operating system and packages are up to date. A reboot maybe required depending on available updates.

```
# yum update
# reboot now
```

Some of Snort 3 dependencies will be installed from source. Create a directory that will hold the source code.

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

Some helper packages are installed to aid completing the setup. These packages are not required by Snort and can be removed later.

```
# yum install vim git wget
```

Snort 3 build 244/245 introduced changes to the way Snort 3 is built[1]:

1. Building Snort 3 using autotools support was removed. This means that cmake must be used to compile Snort and the compilation tools `automake`, `libtool`, `autoconf` are no longer required to be installed.
2. The minimum version of cmake required to build Snort 3 is version 3.4.3, up from version 2.8.11. Versions 3.X of cmake are not available in the CentOS base repository, and will be installed from source.
3. TCmalloc support (added in Snort Build 245).

Basic compilation tools installed from the repository: **flex** (`flex`), **bison** (`bison`), **gcc** (`gcc`), **c++** (`gcc-c++`), and **make** (`make`).

```
# yum install flex bison gcc gcc-c++ make
```

Ensure any previous versions of cmake are removed prior to downloading its source code (3.11.2) for installation.

```
# yum remove cmake
```

```
# wget https://cmake.org/files/v3.12/cmake-3.12.0.tar.gz
# tar xf cmake-3.12.0.tar.gz && cd cmake-3.12.0
# ./configure
# make -j 8
# make install
```

After the installation is complete, the cmake binary will be located at `/usr/local/bin/cmake`. If cmake 2.X was previously installed, the environment variable `PATH` must be updated to add `/usr/local/bin` to the path.

```
# which cmake
/usr/local/bin/cmake
# cmake --version
cmake version 3.12.0
```

# 2. Installing Snort 3 Dependencies

## 2.1 Required Dependencies

Snort 3 required dependencies are installed from both the CentOS base repository and packages source code. This is due to the fact that some packages may not be available in the base repository, or if the packages exist, they are maybe old.

The following packages will be installed from the CentOS base repository: **dnet** (`libdnet-devel`), **hwloc** (`hwloc-devel`), **OpenSSL** (`openssl-devel`), **pkgconfig** (`pkgconfig`), **zlib** (`zlib-devel`).

```
# yum install libdnet-devel hwloc-devel openssl-devel zlib-devel pkgconfig
```

The following dependencies will be installed from their respective source code while demonstrating alternative installation methods when applicable: **LuaJIT**, **pcre**, **pcap**, **daq**.

**LuaJIT**

LuaJIT is used for Snort configuration, optional script plugins for loggers, rule options, and Open AppID detectors. CentOS base repository does not contain the `luajit-devel` package.

To install LuaJIT (2.0.5) from source:

[1] http://blog.snort.org/2018/03/snort-update.html

```
# wget http://luajit.org/download/LuaJIT-2.0.5.tar.gz
# tar xf LuaJIT-2.0.5.tar.gz && cd LuaJIT-2.0.5
# make && make install
# cp /usr/local/lib/pkgconfig/luajit.pc /usr/lib64/pkgconfig/
```

Alternatively, LuaJIT (2.0.4) is available via the EPEL repository, which requires adding the EPEL repository first.

```
# yum install epel-release && yum install luajit-devel
```

### PCRE

The pcre package (8.32) in the base repository, while compatible with Snort 3, is older than the latest version (8.42), and some of Snort 3 optional requirements, Hyperscan, warns that pcre version 8.41 is not installed.

To install PCRE (8.42) from source:

```
# wget https://ftp.pcre.org/pub/pcre/pcre-8.42.tar.gz
# tar xf pcre-8.42.tar.gz && cd pcre-8.42
# ./configure --libdir=/usr/lib64 --includedir=/usr/include
# make && make install
```

Alternatively, to install PCRE from the base repository and ignore Hyperscan warnings:

```
# yum install pcre-devel
```

### PCAP

The **pcap** package (1.5.3) in the base repository is compatible with Snort 3, but older than the latest version (1.9.0).

To install PCAP (1.9.0) from source:

```
# wget http://www.tcpdump.org/release/libpcap-1.9.0.tar.gz
# tar xf libpcap-1.9.0.tar.gz && cd libpcap-1.9.0
# ./configure --libdir=/usr/lib64 --includedir=/usr/include
# make && make install
```

Alternatively, to install PCAP (1.5.3) from the repository: `# yum install libpcap-devel`

### DAQ

Snort 3 requires daq version 2.2.2 for packet IO (afpacket, nfq, etc.). Note that using the nfq module requires installing **libnetfilter** (`libnetfilter_queue-devel`) prior to configuring daq

```
# yum install libnfnetlink-devel libnetfilter_queue-devel
```

Download and decompress daq:

```
# wget https://snort.org/downloads/snortplus/daq-2.2.2.tar.gz
# tar xf daq-2.2.2.tar.gz && cd daq-2.2.2
```

Some of the daq modules can be disabled if not used. The example below configures daq for afpacket while disabling other modules and enabling IPv6 support:

```
# ./configure --disable-ipfw-module --disable-ipq-module --disable-nfq-module --enable-ipv6
```

```
Build AFPacket DAQ module.. : yes
Build Dump DAQ module...... : yes
Build IPFW DAQ module...... : no
Build IPQ DAQ module....... : no
Build NFQ DAQ module....... : no
Build PCAP DAQ module...... : yes
Build netmap DAQ module.... : no
```

Example - Configuring daq for nfq while disabling other modules and enabling IPv6 support:

```
# ./configure --disable-ipfw-module --disable-ipq-module --disable-afpacket-module --enable-ipv6
```

```
Build AFPacket DAQ module.. : no
Build Dump DAQ module...... : yes
Build IPFW DAQ module...... : no
Build IPQ DAQ module....... : no
Build NFQ DAQ module....... : yes
Build PCAP DAQ module...... : yes
Build netmap DAQ module.... : no
```

Proceed with installing DAQ.

```
# make
# make install
```

## 2.2 Optional Dependencies

Snort optional dependencies include: **lzma** (`xz-devel`), **hyperscan**, **cpputest**, **flattbuffers**, **safec**, **uuid** (`uuid-devel`), **iconv**, and **tcmalloc** (`gperftools-libs` and `gperftools-devel`). Some of these are installed from source.

### LZMA and UUID

Lzma is used for decompression of SWF and PDF files. In Snort 2.9.x, this was utilized by the http_inspect preprocessor. Snort 3 requires lzma version >= 5.1.2. The lzma package in CentOS repository is version 5.2.2. Uuid is a library for generating/parsing Universally Unique IDs for tagging/identifying objects across a network.

```
# yum install xz-devel libuuid-devel
```

### Hyperscan

Hyperscan is a high-performance multiple regex matching library. Snort 3 can utilize hyperscan to build new regex and sd_pattern rule options and hyperscan search engine. Prior to installing hyperscan, the following dependencies should be installed or made available: **Ragel**, **Boost,** and **sqlite3** (`sqlite-devel`).

```
# yum install sqlite-devel
```

Download and install Ragel:

```
# wget http://www.colm.net/files/ragel/ragel-6.10.tar.gz
# tar xf ragel-6.10.tar.gz && cd ragel-6.10
# ./configure
# make && make install
```

Download and decompress Boost 1.67 without installation (version 1.68 will cause hyperscan to fail compilation):

```
# wget https://dl.bintray.com/boostorg/release/1.67.0/source/boost_1_67_0.tar.gz
# tar xf boost_1_67_0.tar.gz
```

Download and install Hyperscan:

```
# wget https://github.com/intel/hyperscan/archive/v5.0.0.tar.gz -O hyperscan-5.0.0.tar.gz
# tar xf hyperscan-5.0.0.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_67_0/boost ~/sources/hyperscan-5.0.0/include/boost
# cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local ../hyperscan-5.0.0
```

Method 2 – `BOOST_ROOT`:

```
# cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local -DBOOST_ROOT=../boost_1_67_0
../hyperscan-5.0.0
```

Proceed with installing Hyperscan – using "`-j 8`" will use makefiles in parallel and fasten the make process.

```
# make -j 8
# make install
# cp /usr/local/lib64/pkgconfig/libhs.pc /usr/lib64/pkgconfig/
```

### Cpputest

```
# wget https://github.com/cpputest/cpputest/releases/download/v3.8/cpputest-3.8.tar.gz
# tar xf cpputest-3.8.tar.gz && cd cpputest-3.8
# ./configure --libdir=/usr/lib64 --includedir=/usr/include
# make && make install
```

### Flatbuffers

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

```
# wget https://github.com/google/flatbuffers/archive/v1.9.0.tar.gz -O flatbuffers-1.9.0.tar.gz
# tar xf flatbuffers-1.9.0.tar.gz
# mkdir fb-build && cd fb-build
# cmake ../flatbuffers-1.9.0
# make -j 8 && make install
```

### Safec

Safec is hosted on Sourceforge and some of the mirrors followed by the direct download link may be broken. If the download hangs longer than expected, switch to a different mirror.

```
# wget https://downloads.sourceforge.net/project/safeclib/libsafec-10052013.tar.gz
# tar xf libsafec-10052013.tar.gz
# cd libsafec-10052013
# ./configure --libdir=/usr/lib64 --includedir=/usr/include
# make && make install
```

### Iconv

Iconv is used for converting UTF16-LE filenames to UTF8.

```
# wget https://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.15.tar.gz
# tar xf libiconv-1.15.tar.gz && cd libiconv-1.15
# ./configure
# make && make install
```

### 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 may reduce memory usage. Note that the **gperftools** (`gperftools-devel`) package in the base repository (2.6.1) is older than the latest release (2.7).

To install gperftools (2.7) from source, install the **libunwind** (`libunwind-devel`) package first:

```
# yum install libunwind-devel
```

```
# wget 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 --libdir=/usr/lib64 --includedir=/usr/include
# make && make install
```

Alternatively, to install gperftools (2.6.1) from the repository:

```
# yum install gperftools-libs gperftools-devel
```

## 3. Installing and Verifying Snort 3 Installation

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 configuring Snort with the `configure_cmake.sh` script, set the `LD_LIBRARY_PATH` as follows:

```
# export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib:/usr/local/lib
```

Compiling Snort with tcmalloc support is achieved by passing the `--enable-tcmalloc` argument to the configure command.

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

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

Install options:
    prefix:     /usr/local/snort
    includes:   /usr/local/snort/include/snort
    plugins:    /usr/local/snort/lib64/snort

Compiler options:
    CC:             /bin/cc
    CXX:            /bin/c++
    CFLAGS:            -fvisibility=hidden   -DNDEBUG -g -ggdb  -fno-builtin-malloc -fno-builtin-calloc -fno-builtin-realloc -fno-builtin-free
    CXXFLAGS:          -fvisibility=hidden   -DNDEBUG -g -ggdb  -fno-builtin-malloc -fno-builtin-calloc -fno-builtin-realloc -fno-builtin-free
    EXE_LDFLAGS:
    MODULE_LDFLAGS:

Feature options:
    Flatbuffers:    ON
    Hyperscan:      ON
    ICONV:          ON
    LZMA:           ON
    SafeC:          OFF
    TCMalloc:       ON
    UUID:           ON
-------------------------------------------------------
```

With Snort 3 build 246/247, the configure command no longer fails with errors related to iconv. However, if the iconv errors are encountered, then add `--define=ICONV_ACCEPTS_NONCONST_INPUT:BOOL=true` argument to the configuration command.

Proceed with installing Snort 3.

```
# cd build/
# make -j 8
# make install
```

Once the installation is complete, verify that Snort 3 binary is referencing the expected libraries. Note that Snort 3 binary references the libsfaec library; however, the feature is reported OFF by the cmake configuration summary.

```
# ldd /usr/local/snort/bin/snort
```

```
        linux-vdso.so.1 =>  (0x00007fff3f9cd000)
        libtcmalloc.so.4 => /lib64/libtcmalloc.so.4 (0x00007f453605b000)
        libsfbpf.so.0 => /usr/local/lib/libsfbpf.so.0 (0x00007f4535e34000)
        libpcap.so.1 => /lib64/libpcap.so.1 (0x00007f4535bef000)
        libnfnetlink.so.0 => /lib64/libnfnetlink.so.0 (0x00007f45359e8000)
        libnetfilter_queue.so.1 => /lib64/libnetfilter_queue.so.1 (0x00007f45357e1000)
        libdnet.so.1 => /lib64/libdnet.so.1 (0x00007f45355d0000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f45353cc000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f45351b0000)
        libhwloc.so.5 => /lib64/libhwloc.so.5 (0x00007f4534f73000)
        liblzma.so.5 => /lib64/liblzma.so.5 (0x00007f4534d4d000)
        libluajit-5.1.so.2 => /usr/local/lib/libluajit-5.1.so.2 (0x00007f4534add000)
        libcrypto.so.10 => /lib64/libcrypto.so.10 (0x00007f453467c000)
        libpcre.so.1 => /lib64/libpcre.so.1 (0x00007f453445e000)
        libsafec-1.0.so.1 => /lib64/libsafec-1.0.so.1 (0x00007f4534253000)
        libuuid.so.1 => /lib64/libuuid.so.1 (0x00007f453404e000)
        libz.so.1 => /lib64/libz.so.1 (0x00007f4533e38000)
        libiconv.so.2 => /usr/local/lib/libiconv.so.2 (0x00007f4533b52000)
        libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007f453384b000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f4533549000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f4533333000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f4532f66000)
        libunwind.so.8 => /lib64/libunwind.so.8 (0x00007f4532d4c000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f4536450000)
        libibverbs.so.1 => /lib64/libibverbs.so.1 (0x00007f4532b37000)
        libmnl.so.0 => /lib64/libmnl.so.0 (0x00007f4532931000)
        libnuma.so.1 => /lib64/libnuma.so.1 (0x00007f4532725000)
        libltdl.so.7 => /lib64/libltdl.so.7 (0x00007f453251b000)
        libnl-route-3.so.200 => /lib64/libnl-route-3.so.200 (0x00007f45322ae000)
        libnl-3.so.200 => /lib64/libnl-3.so.200 (0x00007f453208d000)
```

Verify that Snort 3 reports the expected version and library names:

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

```
   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.0.0 (Build 247) 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 2.0.5
           Using OpenSSL 1.0.2k-fips  26 Jan 2017
           Using libpcap version 1.9.0-PRE-GIT (with TPACKET_V3)
           Using PCRE version 8.42 2018-03-20
           Using ZLIB version 1.2.7
           Using FlatBuffers 1.9.0
           Using Hyperscan version 5.0.0 2018-08-29
           Using LZMA version 5.2.2
```

# 4. Installing Snort 3 Extra Plugins 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
# ./configure_cmake.sh --prefix=/usr/local/snort/extra
# cd build/
# make -j 8
# make install
```

# 5. 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.

## 5.1 Global Paths for Rules, AppID, and IP Reputation

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/{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[2].

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.

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

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

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/
```

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

```
# cp etc/* /usr/local/snort/etc/snort/
```

**OpenAppID**

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

```
# wget https://www.snort.org/downloads/openappid/8373 -O snort-openappid-8373.tar.gz
# tar xf snort-openappid-8373.tar.gz
# mv odp/ /usr/local/snort/appid/
```

**IP Reputation**

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.

```
# wget 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'
```

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

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

## 5.2 Setting up 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 ]]
```

## 5.3 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
    ]]
}
```

## 5.4 reputation Inspector

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'
}
```

## 5.5 appid Inspector

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
}
```

## 5.6 file_id and file_log Inspectors

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 over a supported protocol.

The final step is to enable event logging for the inspector. This is accomplished with the `file_log` inspector at the end of the configuration file. 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
}
```

## 5.7 data_log Inspector

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, the inspector must be defined in `snort.lua`. The below example configuration 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
}
```

## 5.8 logger Module

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
```

# 6. 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
```

## 6.1 Running against a PCAP

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
```

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

```
--------------------------------------------------
o")~   Snort++ 3.0.0-247
--------------------------------------------------
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: 10987
               text rules: 10504
            builtin rules: 483
            option chains: 10987
            chain headers: 379
------------------------------------------------------
```

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
```

## 6.2 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. References

- https://www.snort.org/downloads/snortplus/snort_manual.html
- https://github.com/snortadmin/snort3/tree/master/doc