Snort 3 on FreeBSD 11

This guide walks through installing and configuring Snort 3 on FreeBSD 11. 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 FreeBSD image:

```
Base Image      : FreeBSD-11.1-RELEASE-amd64-disc1.iso
Release         : 11.1-RELEASE-p8
Kernel          : 11.1-RELEASE-p8
```

Snort 3 information:

```
Build           : 244
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 package repositories are up to date. Depending on the updates, a reboot maybe required.

```
# freebsd-update fetch
# freebsd-update install
# pkg update
# pkg upgrade
# reboot now
```

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

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

Some helper packages are installed to aid completing the setup. These packages are not required by Snort and can be removed later. Note that install `git` will also install `pcre` (8.40) since it is a dependency to the `git` package.

```
# pkg install git
```

Snort 3 build 244 introduced two major changes to the way Snort 3 is built (http://blog.snort.org/2018/03/snort-update.html):

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. Cmake (3.10.1) is available in the FreeBSD repository, so compiling cmake from source is not required.

Basic compilation tools installed from the repository: **flex** (`flex`), **bison** (`bison`), **gcc** (`gcc`), and **cmake** (`cmake`).

```
# pkg install flex bison gcc cmake
```

2. Installing Snort 3 Dependencies

2.1 Required Dependencies

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

The following packages will be installed from FreeBSD repository: **dnet** (`libdnet`), **hwloc** (`hwloc`), **OpenSSL** (`openssl`), **pkgconfig** (`pkgconf`), **zlib** (`part of the base OS`), **pcre** (`pcre`), **lua** (`luajit` and `lua51`).

```
# pkg install libdnet libpcap hwloc openssl luajit lua51 pkgconf
```

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

PCRE

The pcre package (8.40) in the FreeBSD repository, while compatible with Snort 3, is older than the latest version (8.41), and some of Snort 3 optional requirements, specifically Hyperscan, warns that pcre version 8.41 is not installed.

To install PCRE (8.41) from source:

```
# fetch https://ftp.pcre.org/pub/pcre/pcre-8.41.tar.gz
# tar xf pcre-8.41.tar.gz && cd pcre-8.41
# ./configure
# make && make install
# cp /usr/local/lib/pkgconfig/libpcre*.pc /usr/libdata/pkgconfig/
# ldconfig
```

Alternatively, to install PCRE (8.40) from the repository and ignore Hyperscan warnings:

```
# pkg install pcre
```

DAQ

Snort 3 requires daq version 2.2.2 for packet IO. Some of the daq modules can be disabled if not used.

```
# fetch https://snort.org/downloads/snortplus/daq-2.2.2.tar.gz
# tar xf daq-2.2.2.tar.gz && cd daq-2.2.2
# ./configure --disable-ipq-module --disable-nfq-module --disable-afpacket-module
```

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

Proceed with installing DAQ.

```
# make
# make install
```

2.2 Optional Dependencies

Snort optional dependencies include: **lzma** (`lzlib`), **hyperscan** (`hyperscan`), **cpputest** (`cpputest`)**, flattbuffers** (`flatbuffers`), **safec**, **uuid** (`e2fsprogs-libuuid`), and **iconv** (`libiconv`). Most of these dependencies are available in the FreeBSD repository. Installation using both methods are included in this guide for completeness.

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 library was installed as part of the lzlib package during the installation of the required dependecies. Uuid is a library for generating/parsing Universally Unique IDs for tagging and identifying objects across a network.

Unlike Linux distros, the hyperscan (4.6.0) package is available on the FreeBSD repository, which is one version behind the current release (4.7.0). Installing from repository should reduce the time required to install and maintain the hyperscan package.

The packages **lzma** (`lzlib`), **Hyperscan** (`hyperscan`), **cpputest** (`cpputest`) , **flattbuffers** (`flatbuffers`), **uuid** (`e2fsprogs-libuuid`), and **iconv** (`libiconv`) are installed from the FreeBSD repository. The package **safec** will be installed from source since it does not exist in the FreeBSD repository.

```
# pkg install hyperscan cpputest flatbuffers libiconv lzlib e2fsprogs-libuuid
```

Alternatively, installation from source can proceed as follows (e2fsprogs-libuuid is still installed via pkg).

Hyperscan

Installation from source can be proceed by first installing the Hyperscan dependencies (**Ragel**, **Boost,** and the optional dependency: **sqlite3** (`sqlite3`) and then compiling Hyperscan source.

Download Install Ragel:

```
# fetch 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 Boost, no installation is need:

```
# fetch https://dl.bintray.com/boostorg/release/1.66.0/source/boost_1_66_0.tar.gz
# tar xf boost_1_66_0.tar.gz
```

Install sqlite3:

```
# pkg install sqlite3
```

Download and Install Hyperscan:

```
# fetch https://github.com/intel/hyperscan/archive/v4.7.0.tar.gz -o hyperscan-4.7.0.tar.gz
# tar xf hyperscan-4.7.0.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 the BOOST_ROOT to cmake.

Method 1 – Symlink:

```
# ln -s ~/sources/boost_1_66_0/boost ~/sources/hyperscan-4.7.0/include/boost
# cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local ../hyperscan-4.7.0
```

Method 2 – BOOST_ROOT

```
# cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local -
DBOOST_ROOT=../boost_1_66_0 ../hyperscan-4.7.0
```

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

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

Cpputest

```
# fetch 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.tar.gz
# ./configure
# make && make install
# cp /usr/local/lib/pkgconfig/cpputest.pc /usr/libdata/pkgconfig/
```

Flatbuffers

Flatbuffers is an efficient cross platform serialization library for games and other memory constrained apps. It allows direct access of serialized data without unpacking/parsing it first.

```
# fetch https://github.com/google/flatbuffers/archive/v1.8.0.tar.gz -o flatbuffers-1.8.0.tar.gz
# tar xf flatbuffers-1.8.0.tar.gz
# mkdir fb-build && cd fb-build
# cmake ../flatbuffers-1.8.0
# make && 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.

Currently, compiling safec on FreeBSD may fail and generate an error. This error can be ignored and installation can proceed without installing safec.

```
# fetch https://downloads.sourceforge.net/project/safeclib/libsafec-10052013.tar.gz
# tar xf libsafec-10052013.tar.gz
# cd libsafec-10052013
# ./configure
# make && make install
```

```
In file included from ../include/safe_lib.h:58:0,
                 from safeclib/safeclib_private.h:91,
                 from safeclib/safe_mem_constraint.c:33:
../include/safe_mem_lib.h:87:16: error: conflicting types for 'memset_s'
 extern errno_t memset_s(void *dest, rsize_t dmax, uint8_t value);
                ^~~~~~~
In file included from safeclib/safeclib_private.h:70:0,
                 from safeclib/safe_mem_constraint.c:33:
/usr/include/string.h:158:9: note: previous declaration of 'memset_s' was here
 errno_t memset_s(void *, rsize_t, int, rsize_t);
         ^~~~~~~
*** Error code 1
Stop.
make[2]: stopped in /root/sources/libsafec-10052013/src
*** Error code 1
Stop.
make[1]: stopped in /root/sources/libsafec-10052013
*** Error code 1
```

Iconv

Iconv is used for converting UTF16-LE filenames to UTF8. The libiconv (1.14) package is available in the FreeBSD repository. Note that iconv may cause Snort 3 compilation errors and can be skipped. Additional details are included in Snort installation sections.

To install libiconv (1.14) from repository:

```
# pkg install libiconv
```

Alternatively, to install libiconv (1.15) from source:

```
# fetch 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
```

Additional Dependencies - Bash

While bash is not listed as a Snort 3 dependency, at a certain step during the make process, bash is invoked to generate the `/usr/local/snort/doc` directory and the text documentation files.

In a fresh FreeBSD image or in scenarios where bash is/may not be installed, the Snort 3 make process will fail with an error: "`env: bash: No such file or directory`". There are two options to overcome this error: 1) patch the make scripts to disable the generation of the text documents (http://seclists.org/snort/2018/q1/266), or 2) install bash from the FreeBSD repository.

Since the patch is not official or committed, this guide installs bash, which can be removed after the build is complete.

```
# pkg install bash
```

3. Installing and Verifying Snort 3 Installation

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

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

The following command will start configuring Snort with the supplied arguments such as the prefix. Note that the command may fail and generate the error related to iconv as discussed earlier.

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

```
-- Looking for iconv_open
-- Looking for iconv_open - found
-- Performing Test ICONV_COMPILES
-- Performing Test ICONV_COMPILES - Failed
CMake Error at cmake/FindICONV.cmake:130 (MESSAGE):
  Unable to determine iconv() signature
Call Stack (most recent call first):
  cmake/include_libraries.cmake:25 (find_package)
  CMakeLists.txt:17 (include)
-- Configuring incomplete, errors occurred!
```

If the above error is encountered, add `--define=ICONV_ACCEPTS_NONCONST_INPUT:BOOL=true` argument to the configuration command to become:

```
# ./configure_cmake.sh --define=ICONV_ACCEPTS_NONCONST_INPUT:BOOL=true --prefix=/usr/local/snort
```

Once the configuration completes, the configuration summary is displayed indicating the enabled features.

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

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

Compiler options:
    CC:             /usr/bin/cc
    CXX:            /usr/bin/c++
    CFLAGS:            -fvisibility=hidden   -g -ggdb
    CXXFLAGS:          -fvisibility=hidden   -g -ggdb
    EXE_LDFLAGS:
    MODULE_LDFLAGS:

Feature options:
    Flatbuffers:    ON
    Hyperscan:      ON
    ICONV:          OFF
    LZMA:           ON
    SafeC:          OFF
    UUID:           ON
-----------------------------------------------------
…
-- Build files have been written to: /root/sources/snort3/build
```

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.

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

```
libpcap.so.1 => /usr/local/lib/libpcap.so.1 (0x800d8f000)
libsfbpf.so.0 => /usr/local/lib/libsfbpf.so.0 (0x800fdb000)
libdnet.so.1 => /usr/local/lib/libdnet.so.1 (0x801201000)
libthr.so.3 => /lib/libthr.so.3 (0x801411000)
libhwloc.so.5 => /usr/local/lib/libhwloc.so.5 (0x801639000)
liblzma.so.5 => /usr/lib/liblzma.so.5 (0x801867000)
libluajit-5.1.so.2 => /usr/local/lib/libluajit-5.1.so.2 (0x801a90000)
libcrypto.so.9 => /usr/local/lib/libcrypto.so.9 (0x801e00000)
libpcre.so.1 => /usr/local/lib/libpcre.so.1 (0x802277000)
libuuid.so.1 => /usr/local/lib/libuuid.so.1 (0x802494000)
libz.so.6 => /lib/libz.so.6 (0x802698000)
libflatbuffers.so.1 => /usr/local/lib/libflatbuffers.so.1 (0x8028b1000)
libhs.so.4 => /usr/local/lib/libhs.so.4 (0x802c00000)
libc++.so.1 => /usr/lib/libc++.so.1 (0x803269000)
libcxxrt.so.1 => /lib/libcxxrt.so.1 (0x803530000)
libm.so.5 => /lib/libm.so.5 (0x80374e000)
libgcc_s.so.1 => /lib/libgcc_s.so.1 (0x803979000)
libc.so.7 => /lib/libc.so.7 (0x803b87000)
libpciaccess.so.0 => /usr/local/lib/libpciaccess.so.0 (0x803f3f000)
libxml2.so.2 => /usr/local/lib/libxml2.so.2 (0x804146000)
```

Verify that Snort 3 reports version and the used libraries successfully – Optional dependencies were installed via pkg.

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

```
   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.0.0 (Build 244) FreeBSD
   ''''    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 1.0.2n  7 Dec 2017
           Using libpcap version 1.8.1
           Using PCRE version 8.40 2017-01-11
           Using ZLIB version 1.2.11
           Using FlatBuffers 1.8.0
           Using Hyperscan version 4.6.0 2018-01-03
           Using LZMA version 5.2.3
```

For comparison, below is the output of running the same command after installing the optional dependencies from source (note the reported versions of pcre and hyperscan).

```
  ,,_     -*> Snort++ <*-
 o"  )~   Version 3.0.0 (Build 244) FreeBSD
  ''''    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 1.0.2n  7 Dec 2017
          Using libpcap version 1.8.1
          Using PCRE version 8.41 2017-07-05
          Using ZLIB version 1.2.11
          Using FlatBuffers 1.8.0
          Using Hyperscan version 4.7.0 2018-03-18
          Using LZMA version 5.2.3
```

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 of network traffic decoding, inspection, actions, and logging. One particular Snort 3 extra 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.

Since Snort 3 was cloned from GitHub, the `extra/` directory containing the plugins source code is already available. The prefix to install the extra plugins will follow Snort's initial installation prefix `/usr/local/snort/extra/`.

Before compiling the extra plugins, the environment variable `PKG_CONFIG_PATH` must be set, which also happens to be operating system (64-bit) dependent in build 244. The path can be verified by simply listing Snort 3 installation prefix directory.

```
# cd snort3/extra
# setenv PKG_CONFIG_PATH /usr/local/snort/lib/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, default 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 default and global configuration values of Snort 3 modules.

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 3 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 (http://blog.snort.org/2018/02/snort-30-ruleset-announcement.html).

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 configurations files in from Snort 3 source code 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.

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

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 the 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 created earlier.

```
# fetch https://www.snort.org/downloads/openappid/6329 -o snort-openappid-6329.tar.gz
# tar xf snort-openappid-6329.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.

```
# fetch 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 IP blacklist in the following section.

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

Edit the `snort_defaults.lua` file with your favorite editor. 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'
```

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 mode governs Snort's operational mode, which includes 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 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 preprocessor 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 = "E65ECCC561CACE1860638CD0BC745E59058F16349F7455E215BDDF32333550O7" }, 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

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.

```
# setenv LUA_PATH /usr/local/snort/include/snort/lua/\?.lua\;\;
# setenv 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 will display loaded modules and inspectors, status of parsing reputation lists, and the loaded Snort rules and their counts.

```
--------------------------------------------------
o")~   Snort++ 3.0.0-244
--------------------------------------------------
Loading /usr/local/snort/etc/snort/snort.lua:
        ssh
        .....
    Processing blacklist file /usr/local/snort/etc/snort/../../intel/ip-blacklist
    Reputation entries loaded: 1468, 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: 10374
                text rules: 9897
             builtin rules: 477
             option chains: 10374
             chain headers: 365
--------------------------------------------------
```

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 pass through remaining inspectors for further processing.

With reputation blacklist

```
appid

                    packets: 2875
            processed_packets: 2875
               total_sessions: 4
              firefox_clients: 2
                   http_flows: 2
                   smtp_flows: 1
          thunderbird_clients: 1
-------------------------------------------


data_log
                    packets: 2
-------------------------------------------------

reputation
                    packets: 4
                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: 2875
               total_sessions: 4
              firefox_clients: 2
                   http_flows: 2
                   smtp_flows: 1
          thunderbird_clients: 1
                   icmp_flows: 1
-------------------------------------------


data_log
                    packets: 2
-------------------------------------------------

reputation
                    packets: 4

-------------------------------------------------

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
```

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 events from the text-based rules. The `reputation` inspector generated an alert against the 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 the detected apps and protocols and their associated statistics. From the statistics, Snort was able to detect the use of Firefox and Thunderbird applications and the protocols HTTP and SMPT used to transfer the files.

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

```
1520445690,Firefox,62622,1418464
1520445690,HTTP,62622,1418464
1520445690,SMTP,335494,16292
1520445690,Thunderbird,335494,16292
```

The `file.log` file contains events generated by the `file_id` inspector. The events match the configured `file_policy` to detect and log PDF files and the SHA256 hash of one of the files. Note that the first 2 events detects PDF over SMTP and HTTP respectively. The last event is generated because of detecting the SHA256 of the file transferred 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]

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]

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]
```

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 IP address and port, destination IP address and port, server 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 IP address and port, destination IP address and port, 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 network interface to capture from.

```
# /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