Snort 3.1.6.0 on Ubuntu 18 & 20

Configuring a Full NIDS & SIEM

Noah Dietrich

2021-07-31

Contents

Introduction

This guide shows you how to setup Snort 3 with Splunk as a complete Network Intrusion Detection System (NIDS) and security information and event management (SIEM) system on Ubuntu. The purpose of this guide is to teach you about the components and options that make up Snort and Splunk based NIDS and SIEM so that you can modify Snort and Splunk to meet your specific needs. You can install Snort and Splunk by copying and pasting the individual steps in this guide without taking the time to understand what you are doing, and that will work fine. If however you take the time to understand why you are performing each step, you should have a much deeper understanding of how both Snort and Splunk work.

About Snort 3: Snort 3 is rule-based network intrusion detection and prevention software (NIDS/NIPS).

About Splunk: Splunk is a security information and event management (SIEM) system that collects, stores, and allows you to easily analyze and visualize data, including the alerts created by Snort.

About PulledPork: PulledPork or PulledPork3 is used to download and merge rulesets (the collection of signatures that Snort uses to match against malicious traffic).

About OpenAppID: Snort OpenAppID allows Snort to identify, control, and measure the applications in use on the network. OpenAppID consists of a set of packages (signatures) that match specific types of network data, including layer 7 applications, such as Facebook, DNS, netflix, discus, and google, as well as the applications that use these services (chrome, http, https, etc.).

Software Requirements: This guide has been tested on the 64-bit LTS versions of Ubuntu server 18 and 20. This guide has been tested against Snort 3.1.6.0.

Support: Please ask for help on one of the Snort distribution lists:

- Snort Users
- Snort OpenAppID
- Snort Developers

Most requests should be sent to the **Snort Users** list, unless specifically related to OpenAppID or issues with the codebase. Please read how to ask a good question and understand the mailing list etiquette.

Feedback: Please provide all feedback for this guide, including problems and recommendations to Noah@SublimeRobots.com.

Installing Snort

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

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

Make sure your system has the correct time and the correct time zone. This will be important later when we start processing alerts with Splunk. The command below will allow you to choose your time zone:

```
sudo dpkg-reconfigure tzdata
```

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

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

Install the Snort 3 prerequisites:

```
sudo apt-get install -y build-essential autotools-dev libdumbnet-dev libluajit-5.1-dev libpcap-dev \
    zlib1g-dev pkg-config libhwloc-dev cmake liblzma-dev openssl libssl-dev cpputest libsqlite3-dev \
    libtool uuid-dev  git autoconf bison flex libcmocka-dev libnetfilter-queue-dev libunwind-dev \
    libmnl-dev ethtool
```

Download and install safec for runtime bounds checks on certain legacy C-library calls:

```
cd ~/snort_src
wget https://github.com/rurban/safeclib/releases/download/v02092020/libsafec-02092020.tar.gz
tar -xzvf libsafec-02092020.tar.gz
cd libsafec-02092020.0-g6d921f
./configure
make
sudo make install
```

Snort 3 uses Hyperscan for fast pattern matching. You can install an older version Hyperscan from the Ubuntu repositories, however since Hyperscan is so critical to Snort's operation and performace, it's better to compile the latest stable version of Hyperscan. Hyperscan has a number of reqiurements, including PCRE, gperftools, ragel, and the Boost Libraries.

First Install PCRE: Perl Compatible Regular Expressions. We don't use the Ubuntu repository because it has an older version:

```
cd ~/snort_src/
wget https://ftp.pcre.org/pub/pcre/pcre-8.45.tar.gz
tar -xzvf pcre-8.45.tar.gz
cd pcre-8.45
./configure
make
sudo make install
```

Download and install gperftools 2.9:

```
cd ~/snort_src
wget https://github.com/gperftools/gperftools/releases/download/gperftools-2.9.1/gperftools-2.9.1.tar.gz
tar xzvf gperftools-2.9.1.tar.gz
cd gperftools-2.9.1
./configure
make
sudo make install
```

Download and install Ragel:

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

And finally, download (but don't install) the Boost C++ Libraries:

```
cd ~/snort_src
wget https://boostorg.jfrog.io/artifactory/main/release/1.76.0/source/boost_1_76_0.tar.gz
tar -xvzf boost_1_76_0.tar.gz
```

Install Hyperscan 5.4 from source, referencing the location of the Boost source directory:

```
cd ~/snort_src
wget https://github.com/intel/hyperscan/archive/refs/tags/v5.4.0.tar.gz
tar -xvzf v5.4.0.tar.gz

mkdir ~/snort_src/hyperscan-5.4.0-build
cd hyperscan-5.4.0-build/

cmake -DCMAKE_INSTALL_PREFIX=/usr/local -DBOOST_ROOT=~/snort_src/boost_1_76_0/ ../hyperscan-5.4.0

make
sudo make install
```

Install flatbuffers:

```
cd ~/snort_src
wget https://github.com/google/flatbuffers/archive/refs/tags/v2.0.0.tar.gz -O flatbuffers-v2.0.0.tar.gz
tar -xzvf flatbuffers-v2.0.0.tar.gz
mkdir flatbuffers-build
cd flatbuffers-build
cmake ../flatbuffers-2.0.0
make
sudo make install
```

Next, download and install Data Acquisition library (DAQ) from the Snort website. Note that Snort 3 uses a different DAQ than the Snort 2.9. series. You should check the Snort Website for newer versions of libdaq in case a newer version has been released since this guide was written, or if you get an error that this file is missing.

```
cd ~/snort_src
wget https://github.com/snort3/libdaq/archive/refs/tags/v3.0.4.tar.gz -O libdaq-3.0.4.tar.gz
tar -xzvf libdaq-3.0.4.tar.gz
cd libdaq-3.0.4
./bootstrap
./configure
make
sudo make install
```

Update shared libraries:

```
sudo ldconfig
```

Now we are ready to download, compile, and install Snort 3 from the snort website. If you are interested in enabling additional compile-time functionality, such as the ability to process large (over 2 GB) PCAP files, or the new command line shell: you should run **./configure cmake.sh --help** to list all optional features, and append them to the **./configure_cmake.sh** command below. You should check the Snort Website for newer versions of Snort 3 in case a newer version has been released since this guide was written, or if you get an error that this file is missing.

Download and install, with default settings:

```
cd ~/snort_src
wget https://github.com/snort3/snort3/archive/refs/tags/3.1.6.0.tar.gz -O snort3-3.1.6.0.tar.gz
tar -xzvf snort3-3.1.6.0.tar.gz
cd snort3-3.1.6.0

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

Snort should now be installed under **/usr/local/**. Finally, verify that Snort runs correctly. To do this, we pass the snort executable the **-V** flag (uppercase V for version):

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

You should see output similar to the following:

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

   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.1.6.0
   ''''    By Martin Roesch & The Snort Team
           http://snort.org/contact#team
           Copyright (C) 2014-2021 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using DAQ version 3.0.4
           Using LuaJIT version 2.1.0-beta3
           Using OpenSSL 1.1.1f  31 Mar 2020
           Using libpcap version 1.9.1 (with TPACKET_V3)
           Using PCRE version 8.45 2021-06-15
           Using ZLIB version 1.2.11
           Using FlatBuffers 2.0.0
           Using Hyperscan version 5.4.0 2021-07-05
           Using LZMA version 5.2.4
```

If your output is similar to the above, congratulations! Snort is installed and working.

Now lets test Snort with the default configuration file:

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

You should see output that finishes with the following:

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

Configuring Network Cards

Modern network cards use offloading (LRO for one example) to handle network packet re-assembly in hardware, rather than in software. For most situations this is preferred as it reduces load on the system. For a NIDS, we want to disable LRO and GRO, since this can truncate longer packets (more info in the Snort 2 manual.)

We need to create a systemD service to change these settings. First determine the name(s) of the interfaces you will have snort listen on using **ifconfig**, or if on Ubuntu 20, use the new **ip address show** command.

Once you know the name of the network interface that Snort will listen for traffic on: check the status of large-receive-offload (LRO) and generic-receive-offload (GRO) for those interfaces. In the example below, my interface name is **ens3** (you'll commonly see **eth0** or **ens160** as interface names as well, depending on the system type). We use **ethtool** to check the status:

```
noah@snort3:~$ sudo ethtool -k ens3 | grep receive-offload
generic-receive-offload: on
large-receive-offload: off [fixed]
```

from this output, you can see that GRO is enabled, and LRO is disabled (the 'fixed' means it can not be changed). We need to ensure that both are set to 'off' (or 'off [fixed]'). We could use the ethtool command to disable LRO and GRO, but the setting would not persist across reboots. The solution is to create a systemD script to set this every time the system boots up.

create the systemD script:

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

Enter the following information, replacing **ens3** with your interface name:

```
[Unit]
Description=Ethtool Configration for Network Interface

[Service]
Requires=network.target
Type=oneshot
ExecStart=/sbin/ethtool -K ens3 gro off
ExecStart=/sbin/ethtool -K ens3 lro off

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

Once the file is created, enable and start the service:

```
sudo systemctl enable ethtool
sudo service ethtool start
```

These settings will now persist across reboots. You can verify the setting using ethtool as above.

Configuring Snort

We need to create some folders and files that Snort reqiures for rules:

```
sudo mkdir /usr/local/etc/rules
sudo mkdir /usr/local/etc/so_rules/
sudo mkdir /usr/local/etc/lists/

sudo touch /usr/local/etc/rules/local.rules
sudo touch /usr/local/etc/lists/default.blocklist

sudo mkdir /var/log/snort
```

We will create one rule in the **local.rules** file that you created above:

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

This rule will detect ICMP traffic, and is really good for testing that Snort is working correctly and generating alerts. Paste the following line into the **local.rules** file (make sure that you're copying this line exactly, you must have a space after each semicolon in this file for PulledPork to parse the alert correctly):

```
alert icmp any any -> any any ( msg:"ICMP Traffic Detected"; sid:10000001; metadata:policy security-ips alert; )
```

Now run Snort and have it load the local.rules file (with the **-R** flag) to make sure it loads these rules correctly (verifying the rules are correctly formatted):

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

The output should end with "Snort successfully validated the configuration". You should not have any warnings or errors. If you scroll up through the output, you should see this rule loaded successfully (under the **rule counts** section).

Now let's run Snort in detection mode on an interface (change **eth0** below to match your interface name), and print all alerts to the console:

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

The flags we are using here are:

Flag	Description
-c /usr/local/etc/snort/snort.lua	The snort.lua configuration file.
-R /usr/local/etc/rules/local.rules	The path to the rules file containing our one ICMP rule.
-i eth0	The interface to listen on.
-A alert_fast	Use the alert_fast output plugin to write alerts to the console.
-s 65535	Set the snaplen so Snort doesn't truncate and drop oversized packets.
-k none	Ignore bad checksums, otherwise snort will drop packets with bad checksums

Snort will load the configuration, then display:

```
    Commencing packet processing
    ++ [0] eth0
```

This means that snort is currently listening to all traffic on that interface, and comparing it to the rule it loaded. When traffic matches a rule, Snort will write an alert to the console. Now from another window on that computer (open a new terminal window or a second ssh session), use the ping command to generate packets that traverse the interface you are listening on (ping to that interface's IP address if you're connecting from another computer, or just ping an external ip address if you're on the same machine. You should see alerts print on the screen:

```
    12/15-21:02:26.976073 [**] [1:10000001:0] "ICMP Traffic Detected" [**] [Priority: 0] {ICMP} 10.10.10.1 -> 10.10.10.88
    12/15-21:02:26.976157 [**] [1:10000001:0] "ICMP Traffic Detected" [**] [Priority: 0] {ICMP} 10.10.10.88 -> 10.10.10.1
```

Use **ctrl-c** to stop Snort. This is a good rule for testing Snort, but can be a little noisy during actual production usage so comment it out with the hash symbol if you like.

Next let's edit the **snort.lua** file. This file is the configuration file we pass to Snort at startup:

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

Next, we want to enable decoder and inspector alerts (malicious traffic that is detected by Snort, not the rules due to the more complex format), and we want to tell the ips module where our rules file will be (PulledPork will create this for us later)

Scroll down to line 169, and look for the section titled **ips**. Here we un-comment (remove the leading two dashes) from **enable_builtin_rules=true**, and enable our pulledpork rules. Note that lua uses four spaces, not tabs to indent these lines (this is required). This section should look like this (comments removed):

```
169 ips =
170 {
171     enable_builtin_rules = true,
172     include = RULE_PATH .. "/local.rules",
173     variables = default_variables
174 }
```

test your config file (since we've made changes):

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

Now we can run snort as above, however we don't explictly pass the **local.rules** file on the command line, as we've included it in the **ips** section in the **snort.lua** file:

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

Ping the interface as above, and you should see the alerts written to the console again.

PulledPork

PulledPork is a tool that we will use to download rulesets, which are the latest rules files that snort/talos releases to ensure that your system can detect the latest attacks. The original PulledPork is written in perl, and has worked wonderfully for years supporting Snort 2. With Snort 3, it was decided to re-write PulledPork in Python 3 and not add Snort3 functionality to the original PulledPork.

You need to decide if you want to use the original PulledPork or the new PulledPork3 for ruleset management; each have strenghs and weaknesses. In the near future you'll only want to use PulledPork3, but it's still in Beta and may not be the correct solution right now, depending on your needs.

The original **PulledPork** has not been modified to fully support Snort 3: it will not reload Snort 3 after updating rules, and it does not handle compiled (.SO rules). However, the code base is very stable, and ouside of those two limitations, it is a solid product.

The new **PulledPork3** is still in Beta, and is being activly devloped. It does handle .so rules, reloading Snort, and some of the advanced functionality offered by the originall PulledPork. However: it is still under development, and may break going fowards as bugs are found and features added.

Deciding what version of PulledPork to use: if this is a production system and you can do without the .so rules, can set Snort to reload manually after updates, and you require a stable envirornment: stick with the original PulledPork. If this is a test system, or not mission critical: you can probably use PulledPork3.

Once you've determined which version of PulledPork you want to use, follow those instructions from the correct section below: **PulledPork3** or **PulledPork Original**.

PulledPork3

https://github.com/shirkdog/pulledpork3 is conceptually similar to the original PulledPork, there's a single script to run, and you pass it a configuration file to make sure you're geting the files you want.

You will want to register with snort.org for a free oinkcode. This unique code will allow you to download the "registered" and "LightSPD" rulesets . Without an Oinkcode, you are limited to downloading the "community" rulesets, which do not have many recent rules. You'll need this oinkcode later in the installation, so keep it handy.

Start by obtaining the latest version of PulledPork3:

```
cd ~/snort_src/
git clone https://github.com/shirkdog/pulledpork3.git
```

Next, copy the python file to an apropriate location:

```
cd ~/snort_src/pulledpork3
sudo mkdir /usr/local/bin/pulledpork3
sudo cp pulledpork.py /usr/local/bin/pulledpork3
sudo cp -r lib/ /usr/local/bin/pulledpork3

sudo chmod +x /usr/local/bin/pulledpork3/pulledpork.py

sudo mkdir /usr/local/etc/pulledpork3
sudo cp etc/pulledpork.conf /usr/local/etc/pulledpork3/
```

verify the PulledPork3 runs:

```
/usr/local/bin/pulledpork3/pulledpork.py -V
```

You should see something similar to the following (version numbers may change):

```
noah@snort3:~$ /usr/local/bin/pulledpork3/pulledpork.py -V
PulledPork v3.0.0-BETA

    https://github.com/shirkdog/pulledpork3
      _____ ____
     ----,\    )   PulledPork v3.0.0-BETA
      --==\\  /    Lowcountry yellow mustard bbq sauce is the best bbq sauce. Fight me.
       --==\\/
     .-~~~~~-.Y|\\_  Copyright (C) 2021 Noah Dietrich, Colin Grady, Michael Shirk
  @_/        /  66\_  and the PulledPork Team!
    |    \   \   _(")
     \   /-| ||'--'   Rules give me wings!
      \_\  \_\\
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Next, we need to modify our pulledpork.conf file.

```
sudo vi /usr/local/etc/pulledpork3/pulledpork.conf
```

This configuration file is broken up into sections, and there are a number of options. We'll opt for a simple configuration. We will download the LightSPD ruleset by setting it to **true** (If you chose to not register with snort.org for the free oinkcode, you'll be limited to the community ruleset)):

```
3  community_ruleset = false
4  registered_ruleset = false
5  LightSPD_ruleset = true
```

Enter your oinkcode (line 8) from snort.org if you're using the LightSPD_ruleset.

If you want to download and use blocklists, set one or both of the blocklists to true (lines 12 and 13).

```
12  snort_blocklist = true
13  et_blocklist = true
```

PulledPork needs to know where your snort binary is located. Set the snort_path to point to the snort binary path (and make sure to un-comment this line):

```
30  snort_path = /usr/local/bin/snort
```

Where are your local rules saved (to include in pulledpork.rules)

```
62  local_rules = /usr/local/etc/rules/local.rules
```

Now run PulledPork3:

```
sudo /usr/local/bin/pulledpork3/pulledpork.py -c /usr/local/etc/pulledpork3/pulledpork.conf
```

Once PulledPork3 finishes execution, there are multiple files created for you: **/usr/local/etc/rules/pulledpork.rules** will contain all the rules from the downloaded ruleset, along with the rules from your **local.rules** file. Compiled rules, also called .so rules (referenced by some of the rules in pulledpork.rules) will be saved in **/usr/local/etc/so_rules/**. blocklists (if enabled) will be downloaded, merged together, and written to **/usr/local/etc/lists/default.blocklist**.

We need to modify our **snort.lua** file to load the new pulledpork.rules files (we don't need to modify the blocklist, as that path was already set earlier).

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

Modify the ips include option to change it from your local rules file to the new pulledpork.rules file:

```
ips =
{
    enable_builtin_rules = true,
    include = RULE_PATH .. "/pulledpork.rules",
    variables = default_variables
}
```

Since you've modified the snort.lua, you should test it. We need to add an additional parameter here, to tell Snort3 where the .so rules are located. The pulledpork.rules file contains a number of rules that reference the compiled .so rules, and if it can't find the actual compiled files (in /usr/local/etc/so_rules) you'll get an error.

```
snort -c /usr/local/etc/snort/snort.lua --plugin-path /usr/local/etc/so_rules/
```

Now let's create a systemd scheduled task to have pulledpork update rules daily. Due to the way systemd timers work, this requires two parts. First we create the systemd unit file to runn pulledpork

```
sudo vi /lib/systemd/system/pulledpork3.service
```

enter the following text:

```
[Unit]
Description=Runs PulledPork3 to update Snort 3 Rulesets
Wants=pulledpork3.timer

[Service]
Type=oneshot
ExecStart=/usr/local/bin/pulledpork3/pulledpork.py -c /usr/local/etc/pulledpork3/pulledpork.conf

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

Enable and run the PulledPork unit file. This may take a minute to complete:

```
sudo systemctl enable pulledpork3
sudo service pulledpork3.timer start
```

Now that we have the service file created for pulledpork3, we need to create a systemd timer to run it daily. Create the timer:

```
sudo vi /lib/systemd/system/pulledpork3.timer
```

with the following content:

```
[Unit]
Description=Run PulledPork3 rule updater for Snort 3 rulesets
RefuseManualStart=no  # Allow manual starts
RefuseManualStop=no   # Allow manual stops

[Timer]
#Execute job if it missed a run due to machine being off
Persistent=true
#Run 120 seconds after boot for the first time
OnBootSec=120
#Run daily at 1 am
OnCalendar=*-*-*13:35:00
#File describing job to execute
Unit=pulledpork3.service

[Install]
WantedBy=timers.target
```

This will run pulledpork daily at 13:35 every day, as well as 2 minutes after the sytem powers on. You should choose a different time to not overwhelm Snort.org's servers (run once per day).

Finally, enable the timer

```
sudo systemctl enable pulledpork3.timer
```

PulledPork Original

Follow this section's instructions to install the Orriginal (stable, but with less features) version of PulledPork.

Start by registering an account on the Snort website to get a unique your oinkcode before continuing, as the oinkcode is required for the most popular free ruleset.

Install the PulledPork pre-requisites:

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

Next, download the latest version of PulledPork and install it by copying the perl file to **/usr/local/bin** and the needed configuration files to **/usr/local/etc/pulledpork**:

```
cd ~/snort_src
wget https://github.com/shirkdog/pulledpork/archive/master.tar.gz -O pulledpork-master.tar.gz
tar xzvf pulledpork-master.tar.gz
cd pulledpork-master/

sudo cp pulledpork.pl /usr/local/bin
sudo chmod +x /usr/local/bin/pulledpork.pl

sudo mkdir /usr/local/etc/pulledpork
sudo cp etc/*.conf /usr/local/etc/pulledpork
```

Test that PulledPork runs by running it with the **-V** flag as done below, looking for the output below:

```
noah@snort3:~$ /usr/local/bin/pulledpork.pl -V
PulledPork v0.8.0 - The only positive thing to come out of 2020...well this and take-out liquor!
```

Now that we are sure that PulledPork runs, we need to configure it:

```
sudo vi /usr/local/etc/pulledpork/pulledpork.conf
```

line 19, we need to change the URL, and then replace **<oinkcode>** with the oinkcode you got when you registered with the snort.org website. This tells PulledPork where to download the rules from.

```
19  rule_url=https://www.snort.org/rules/|snortrules-snapshot.tar.gz|<oinkcode>
```

line 21: Comment out the community rules. These are not needed since they are included in the registered ruleset we included above:

```
21  #rule_url=https://snort.org/downloads/community/|community-rules.tar.gz|Community
```

line 72: We need to point ot the correct snort.rules file, where PulledPork will save all the rules it downloads and includes from the local.rules file:

```
72  rule_path=/usr/local/etc/rules/snort.rules
```

line 87: We need to tell PulledPork where the local.rules file is to copy rules from (and into our snort.rules):

```
87  local_rules=/usr/local/etc/rules/local.rules
```

line 94: This tells PulledPork to output metadata about the rules in the newer sid_msg format:

```
94  sid_msg_version=2
```

line 134, change the distro to Ubuntu-18-4 (even if you're running Ubuntu 20):

```
134  distro=Ubuntu-18-4
```

line 142: This tells PulledPork where to save the blocklist (IP Addresses that are known to be malicious and should be blocked):

```
142  block_list=/usr/local/etc/lists/default.blocklist
```

line 151: Tell PulledPork the default location for block and allow lists:

```
151  IPRVersion=/usr/local/etc/lists
```

line 209: uncomment this line to enable all rules in the downloaded rule file. The rules are split into different rulesets, depending on how aggresive you want to detect traffic. If you were running in IPS mode (blocking instead of detecting traffic). You might consider going with the "ballanced" ruleset rather than "security", as the "security" ruleset is more aggressive about detecting traffic that might be malicious, or might be normal:

```
209  ips_policy=ballanced
```

Run PulledPork, passing it our configuration file and do extra logging. This will download the latest rulesets, combine them with any rules in our **local.rules** file and save all the rules into **snort.rules**, as well as save blacklist entries in our defautl.blocklist file:

```
    sudo /usr/local/bin/pulledpork.pl -c /usr/local/etc/pulledpork/pulledpork.conf -l -P -E -T
```

we are using the following flags:

Flag	Description
-c /usr/local/etc/pulledpork/pulledpork.conf	The PulledPork configuration file
-l	log important info to syslog

Flag	Description
-P	Process rules even if no new rules downloaded
-E	only write enabled rules out
-T	Do not use .so rules (they don't work with original PuledPork)

you should see output similar to the following:

```
    Rule Stats...
        New:-------15047
        Deleted:---0
        Enabled Rules:----15048
        Dropped Rules:----0
        Disabled Rules:---0
        Total Rules:------15048
    IP Blocklist Stats...
        Total IPs:-----760

    Done
    Please review /var/log/sid_changes.log for additional details
    Fly Piggy Fly!
```

If this works, the next step is to turn this command into a scheduled task, so that we can update our rulesets daily:

```
    sudo crontab -e
```

The Snort team has asked you to randomize when PulledPork connects to their server to help with load balancing. In the example below, we have PulledPork checking at 13:44 every day. Change the minutes value (the 44 below) to a value between 0 and 59, and the hours value (the 13 below) to a value between 00 and 23:

```
    44 13 * * * /usr/local/bin/pulledpork.pl -c /usr/local/etc/pulledpork/pulledpork.conf -l -P -E -T
```

check your **snort.rules** file, you should see a number of new rules that pulledpork installed for you to use.

Modify **snort.lua** to load the **snort.rules** rather than the **local.rules** file (the rules in our local.rules are automatically added to the **snort.rules** file automatically by PulledPork along with all the downloaded rules, you should see any rules from your **local.rules** included at the end of the **snort.rules** file):

```
167  ips =
168  {
169      enable_builtin_rules = true,
170      include = RULE_PATH .. "/snort.rules",
171      variables = default_variables
172  }
```

Test Snort to see if those rules load correctly:

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

scroll up, and you should see something like:

```
    rule counts
           total rules loaded: 15573
                   text rules: 15048
                builtin rules: 525
                option chains: 15573
                chain headers: 330
```

Configuring Snort Plugins

We want to enable a number of features in our **snort.lua** file:

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

First let's configure our **HOME_NET** variable. This refers to the local subnet we are defending (rules use this information to determine if an alert matches). Set your local subnet information here to match your subnet. My subnet below is the 10.0.0.0 network with a 24-bit subnet mask:

```
24  HOME_NET = '10.0.0.0/24'
```

Enable hyperscan (faster pattern matching): more info here, place this after the repuatation inspector, but before section 3: **configure bindings**:

```
104 search_engine = { search_method = "hyperscan" }
105
106 detection = {
107     hyperscan_literals = true,
108     pcre_to_regex = true
109 }
```

Enable the reputation blocklist. remove the comments (the block comments around the entire reputation block) from the reputation inspector, and enable the blocklist (note, the snort.lua file uses the older, unsupported "blacklist", which has been replaced with blocklist):

```
96  reputation =
97  {
98     blocklist = BLACK_LIST_PATH .. "/default.blocklist",
99  }
```

since you've made changes, you should verify the configuration again:

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

(remember, if you're Using PulledPork3, you need to include the --plugin-path option pointing to the so_rules folder) instead:

```
snort -c /usr/local/etc/snort/snort.lua --plugin-path /usr/local/etc/so_rules/
```

JSON Alerts Output Plugin

In order to easily import the Snort 3 alert log files into your SIEM of choice (like Splunk), you will want to use the **alert_json** output plugin to write all alerts to a json-formatted text file. Enabling the json output plugin is easy, just modify your **snort.lua** file (in section 7: configure outputs, around line number 230):

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

First, enable the **alert_json** plugin as shown below. Remember that indents use 4 spaces instead of a tab:

```
alert_json =
{
    file = true,
    limit = 100,
    fields = 'seconds action class b64_data dir dst_addr dst_ap dst_port eth_dst eth_len \
    eth_src eth_type gid icmp_code icmp_id icmp_seq icmp_type iface ip_id ip_len msg mpls \
    pkt_gen pkt_len pkt_num priority proto rev rule service sid src_addr src_ap src_port \
    target tcp_ack tcp_flags tcp_len tcp_seq tcp_win tos ttl udp_len vlan timestamp',
}
```

In the **alert_json** plugin, we are specifying three options:

1. First we use the **file** option to enable outputting alerts to the json-formatted file (instead of to the console).
2. Next we specify the **limit** option to tell Snort when to roll over to a new file. When the output file reaches 10 MB, a new file will be created, using the current unixtime in the filename. We set this to 100 MB for testing, but on a production system you probably want to increase this number, depending on how you're doing log management / rotation.
3. Finally we specify the **fields** option, which identifies which specific fields from the alert should be included in the json output. In this example we have chosen every possible field to be output.

Note: After testing, you can choose to remove some of these fields (the *vlan* and *mpls* fields are often not necessary, and the *b64_data* contains the entire packet payload, which can be removed to save space, although there is a lot of good info in this field). Do not remove the **seconds** field, and make sure it is always the first field listed. This will allow Splunk to correctly process the events.

Now we want to run Snort, and generate some alerts. these alerts will be written to /var/log/snort. Run the below, and ping the interface again (like we did before to generate traffic that matches the rule in our local.rules file):

```
sudo /usr/local/bin/snort -c /usr/local/etc/snort/snort.lua -s 65535 \
    -k none -l /var/log/snort -i eth0 -m 0x1b
```

(and again, include --plugin-path if using PulledPork3)

We added a few new flags to this command:

Flag	Description
-l var/log/snort	the directory where log files should be written
-m 0x1b	Umask of 033 for file permissions (rw-r–r–)

you won't see anything output to the screen after snort starts, since we've enabled the alert_json output module (which writes to /var/log/snort as specified in the command above). Stop snort (ctrl-c), and then check /var/log/snort:

```
cat /var/log/snort/alert_json.txt
```

you'll see the data from the alert in JSON format in the file:


```
{ "seconds" : 1608147213, "action" : "allow", "class" : "none", "b64_data" : "
    DWHaXwAAAADO0wgAAAAAABAREhMUFRYXGBkaGxwdHh8gISIjJCUmJygpKissLS4vMDEyMzQ1Njc=", "dir" : "S2C", "
    dst_addr" : "10.10.10.1", "dst_ap" : "10.10.10.1:0", "eth_dst" : "52:54:00:1F:8A:1C", "eth_len" :
    98, "eth_src" : "52:54:00:70:78:9F", "eth_type" : "0x800", "gid" : 1, "icmp_code" : 0, "icmp_id" :
    5203, "icmp_seq" : 3, "icmp_type" : 0, "iface" : "ens3", "ip_id" : 3006, "ip_len" : 64, "msg" : "
    ICMP Traffic Detected", "mpls" : 0, "pkt_gen" : "raw", "pkt_len" : 84, "pkt_num" : 8, "priority" :
    0, "proto" : "ICMP", "rev" : 0, "rule" : "1:10000001:0", "service" : "unknown", "sid" : 10000001, "
    src_addr" : "10.10.10.88", "src_ap" : "10.10.10.88:0", "tos" : 0, "ttl" : 64, "vlan" : 0, "
    timestamp" : "12/16-20:33:33.603502" }
```

Snort Startup Script

We create a systemD script to run snort automatically on startup. We will also have snort run as a regular (non-root) user after startup for security reasons. First create the snort user and group:

```
sudo groupadd snort
sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort
```

remove old log files (move them if you want to keep them):

```
sudo rm /var/log/snort/*
```

We need to grant the 'snort' user rights to the log directory:

```
sudo chmod -R 5775 /var/log/snort
sudo chown -R snort:snort /var/log/snort
```

create the systemD service file:

```
sudo vi /lib/systemd/system/snort3.service
```

with the following content (change the ethernet adapter **eth0** to match your adapter):

Note: if you're running PulledPork3, you'll need one additional paramter that will load the .so rules from /usr/local/etc/so_rules/, the **--plugin-path** option.

```
[Unit]
Description=Snort3 NIDS Daemon
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/snort -c /usr/local/etc/snort/snort.lua -s 65535 \
    -k none -l /var/log/snort -D -u snort -g snort -i eth0 -m 0x1b --create-pidfile \
        --plugin-path=/usr/local/etc/so_rules/

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

Here's a breakdown of all the flags we are using with Snort:

Flag	Description
/usr/local/bin/snort	This is the path to the snort binary. We don't use sudo here since the script will be started with elevated (root) privileges.
-c /usr/local/etc/snort/snort.lua	The snort.lua configuration file.
-s 65535	Set the snaplen so Snort doesn't truncate and drop oversized packets.
-k none	Ignore bad checksums, otherwise snort will drop packets with bad checksums, and they won't be evaluated.
-l /var/log/snort	The path to the folder where Snort will store all the log files it outputs.
-D	Run as a Daemon.
-u snort	After startup (and after doing anything that requires elevated privileges), switch to run as the "snort" user.
-g snort	After startup, run as the "snort" group.

Flag	Description
-i eth0	The interface to listen on.
-m 0x1b	Umask of 033 for file permissions.
--create-pidfile	Create a PID file in the log directory (so pulledpork can restart snort after loading new rules)
--plugin-path	For PulledPork3 only: where are the .so rules stored.

Enable the Snort systemD service and start it:

```
sudo systemctl enable snort3
sudo service snort3 start
```

check the status of the service:

```
service snort3 status
```

your output should be similar to the following, showing 'active (running)':

```
noah@snort3:~/pcaps$ service snort3 status
* snort3.service - Snort3 NIDS Daemon
   Loaded: loaded (/lib/systemd/system/snort3.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2018-12-11 16:48:44 EST; 2min 57s ago
```

you can check the full output of the service with the following command if there are any problems:

```
sudo journalctl -u snort3.service
```

If you're running PulledPork3: we can now configure it to tell Snort to reload the rules files and blocklist files when PulledPork3 modifies them.

Edit your pulledpork.conf file:

```
sudo vi /usr/local/etc/pulledpork3/pulledpork.conf
```

Uncomment line 40:

```
40  pid_path=/var/log/snort/snort.pid
```

Now when PulledPork3 runs, it will tell snort the files are modifed, and to reload them.

Splunk

Splunk is the software we will use as our SIEM (Security information and event management) solution, which will display graphically (through a web interface) all the alerts Snort has generated, and will give us some powerful tools to search and understand those alerts, as well as draw deeper information from them. Splunk is free (as in cost) software for the way we are using it (although you can purchase a license for additional functionality relating mostly to managing large Splunk installations). Alternative software would be Elasticstack's ELK stack (which I don't use here because the configuration is more complex).

Installing Splunk:

You will need to create a free account on Splunk's website to download the software and Add-ons. Navigate to Splunk's Homepage, click on the **Free Splunk** button in the upper right, create a new account (or login if you already have an account). Under **Splunk Free** you will click the link under "Splunk Enterprise" titled Download Free 60-day trial.

On the download page, click the **Linux** tab, and then click the **Download Now** button next to **.deb** (since we're running Ubuntu, a Debian-based system). Agree to the license, and click the **start your download now** button. The download page will automatically open up a window to save the download to your local system. If you want to use **wget** to download the installer instead, you can cancel this download, then click **Download via Command Line (wget)** to copy the wget string for your download. The download is approximately 420 MB.

Once you have the Splunk installer on your system, you need to install it. From the directory where you saved the installer:

```
sudo dpkg -i splunk-8.*.deb
sudo chown -R splunk:splunk /opt/splunk
```

This will install Splunk to **/opt/splunk**. Note that the volume Splunk is installed to must have 5 GB of free space or Splunk will not start. The indexes where Splunk stores all the collected log data reside in a sub-folder of the install location, so make sure there's enough space on this volume for all the data you expect to collect.

Now we want to start Splunk for the first time (accepting the license and taking all default options), You will be prompted to create a new admin user and password for Splunk. Save these credentials, as we will use them later to log into the web interface:

```
sudo /opt/splunk/bin/splunk start --answer-yes --accept-license
```

then we want to configure Splunk to start automatically at boot time. We will also enable systemD for Splunk and start the service (not the uppercase "S" in the Splunk systemD service name).

```
sudo /opt/splunk/bin/splunk stop
sudo /opt/splunk/bin/splunk enable boot-start -systemd-managed 1

sudo chown -R splunk:splunk /opt/splunk
sudo service Splunkd start
```

The Splunk server is now listening on port 8000 of this server (http://localhost:8000 if you're connecting from the local machine, or via the IP address of this system from another computer). The username and the password are the ones you setup when installing Splunk.

Splunk is running with the free Enterprise Trial license at this time, giving all Enterprise features for 60 days, and allowing you to index 5 GB of log data per day. The only feature that we will loose once the trial license expires that will affect this installation is the removal of authenticated logins. Once you convert to the free license, you will not be prompted to log into the Splunk web interface.

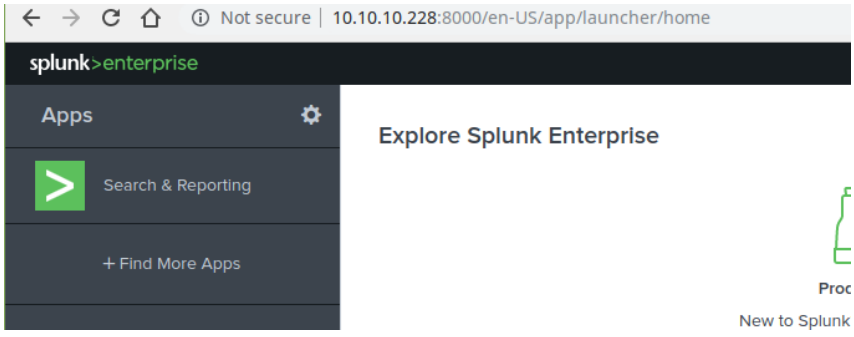
Splunk Enterprise offers a number of features, including a deployment server to automatically update Splunk instances and the Splunk apps they run automatically, multiple user accounts with configurable permissions, load balancing, and other features.

Configuring Splunk:

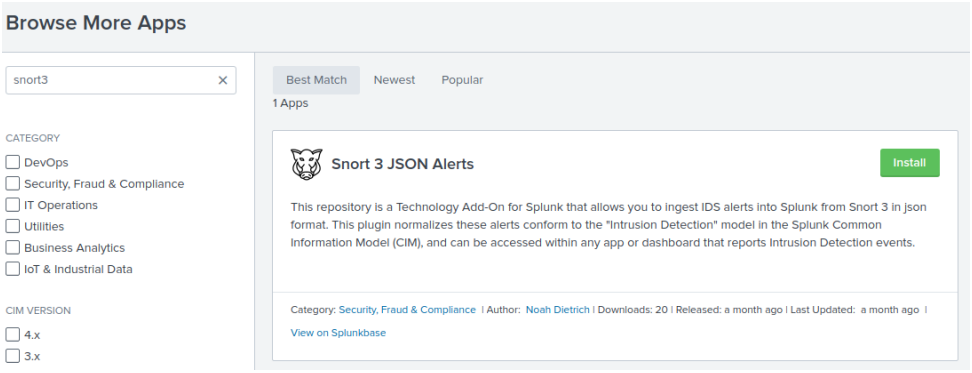
Now log onto your Splunk instance, using the username and password you created during the Splunk install. The Splunk server is listening on port 8000 (http://localhost:8000).

We need to install a Splunk Plugin (called an Add-on) that will allow us to easily ingest (collect) logs created by Snort 3 and normalize them (make sure field naming is consistent with NIDS data so that Splunk apps can display our data easily).

To install this app, from the main web page of your Splunk instance, click the link titled **+Find More Apps** on the left side of the Splunk Web Interface:

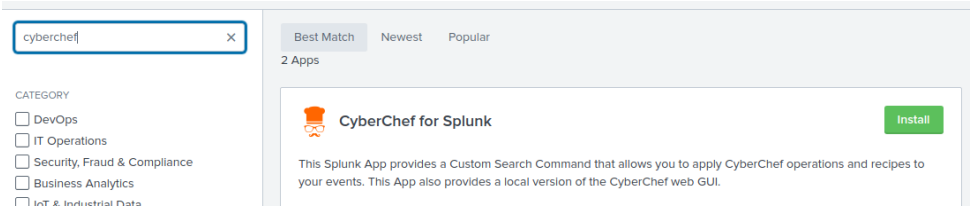


This will take you to Splunkbase, an on line repository for Splunk Add-ons that extend and enhance the functionality of your Splunk installation. Search Splunkbase for **Snort** and you'll be presented with one result: **Snort 3 JSON Alerts**. Click the green **install** button, next to this Add-on:



enter the username and password your created with Splunk when you registered to download Splunk (not the username and password you created for your local Splunk server instance). Accept the terms & conditions, and click **Login and Install**. click **done** once the install is completed.

Next we want to install the **CyberChef for Splunk** plugin that will allow us to covert the b64_data fields into readable text. Just like above, search Splunkbase for "cyberchef", click the greeen **install** button next to **CyberChef for Splunk**, login, and then install:



Next,we need to configure the Snort 3 JSON Alerts add-on to tell Splunk where the log files are stored that Snort 3 generated so Splunk can ingest them. We do this from the command line with a configuration file:

```
sudo mkdir /opt/splunk/etc/apps/TA_Snort3_json/local
sudo touch /opt/splunk/etc/apps/TA_Snort3_json/local/inputs.conf
sudo vi /opt/splunk/etc/apps/TA_Snort3_json/local/inputs.conf
```

Enter the following text into this **inputs.conf** file:

```
[monitor:///var/log/snort/*alert_json.txt*]
sourcetype = snort3:alert:json
```

restart Splunk:

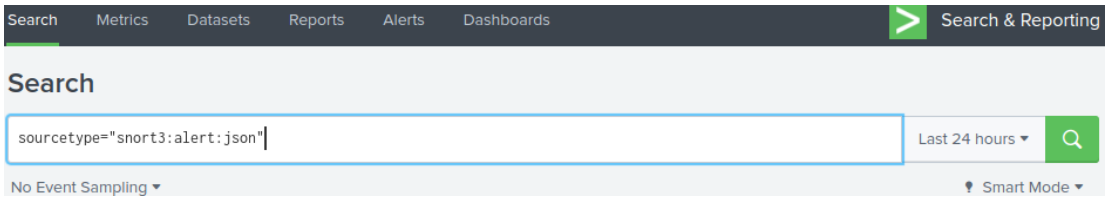
```
sudo service Splunkd restart
```

now when Splunk starts, it will scan the **/var/log/snort** directory for json files, assign them sourcetype of **snort3:alert:json**, and ingest them so we can search them.

from your Splunk instance, log in (since you rebooted the server), click the **Splunk > Enterprise** link in the upper left, and then click the **Search and Reporting** app link on the left side. In the search field, enter the following text:

```
sourcetype="snort3:alert:json"
```

and then click the green magnifying glass icon to start the search.



This will show all events that our server is collecting. you may not see many events, especially if you deleted the old json files we created from our pcap files. You can create a few new alerts using **ping** if you want (remember we created that rule earlier) if you don't see any alerts. There is a slight lag between an event being generated and shown in Splunk. If you continue to not see any alerts, change the time range (the drop-down set to the past 24 hours next to the search icon) to **all time** and re-run the search. If you still don't see any events, check that there are json files in your **/var/log/snort** folder.

Using Splunk

This guide does not go in-depth with using Splunk. There are excellent free resources available from Splunk which I mention below.

Below are some simple searches you may find helpful in starting out. To show all events in a table with the time, source, destination, and message, run the following search:

```
sourcetype="snort3:alert:json"
| table _time src_ap dst_ap msg
```

To show the count of all events by destination:

```
sourcetype="snort3:alert:json"
| stats count by dest
```

to show all events sources on a map:

```
sourcetype="snort3:alert:json"
| iplocation src_addr
| stats count by Country
| geom geo_countries featureIdField="Country"
```

(you may need to click on the "Visualization" tab, and then "line chart" and change it to Choropleth Map)

For many of your events, there will payload data (the b64_data) field that's base64 encoded (http and SMTP are a good example of this). To convert this data so we can read it, we use the "cyberchef" function to conver it for each event (on the fly), and add a new field to each event called "decrypted":

```
1 sourcetype="snort3:alert:json" dest_port=80
2 | cyberchef infield='b64_data' outfield=decrypted operation="FromBase64"
3 | table src_addr, dst_addr, rule, msg, decrypted
```

Some excellent *free* resources for using Splunk are:

EBook: Exploring Splunk: Search Processing Language (SPL) Primer and Cookbook

Free Online Training: - Free Splunk Fundamentals 1 - Splunk Infrastructure Overview

Cleaning up your install

Splunk is currently running in **Free Enterprise Trial Mode**, which is only good for 60 days. We want to convert this license into the **free** mode, which is similar to Enterprise mode, with a few features removed. The feature you will notice missing are the ability to log onto the

server with a username and password (allowing anyone to log on). You also loose some features related to clustering, as well as the ability to deploy Splunk apps to other servers (useful when you have more than one server or system to collect logs from).

To change the license: Click **Settings** along the upper right bar, and then click **Licensing**:

Click **change license group**. Select **Free License** and click **Save**. click **Restart Now**, and click **OK**.

you may now find that you can't access the Spunk web interface from a remote computer, that's ok, and we'll fix that in the next section with a reverse proxy.

Reverse proxy for Splunk Web

Splunk with a free license doesn't prevent access with a username and password, and will only allow access from the local machine (depeding if you were connected locally when you switched licenses). Here we will setup a reverse proxy with apache listening on port 80 of this server, which will reqiure a password, and will redirect to the Splunk interface.

Install apache and the proxy modules:

```
sudo apt-get install -y apache2 apache2-utils

sudo a2enmod proxy
sudo a2enmod proxy_http
sudo systemctl restart apache2
```

create a new user. you'll be prompted for the password. Remember the user and password, as you'll use it later. You can create multiple users this way (replace with an actual username:

```
sudo touch /etc/apache2/.htpasswd
sudo htpasswd /etc/apache2/.htpasswd <username>
```

edit the apache configuration file to setup the proxy listening on port 80:

```
sudo vi /etc/apache2/sites-available/000-default.conf
```

Enter the following information into this file. The file should already exist with some content. Below you can see what this file needs to look like, with the comments remove. These new configuration options are telling apache to listen on port 80, require authentication (the user/password in the htaccess file), and then forward all authorized connections to localhost port 8000 (where splunk is listening).

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined


        # Added section for Splunk Reverse Proxy with authentication
        ProxyPreserveHost On
        ProxyPass / http://127.0.0.1:8000/
        ProxyPassReverse / http://127.0.0.1:8000/
        <Proxy *>
                Order deny,allow
                Allow from all
                Authtype Basic
                Authname "Password Required"
                AuthUserFile /etc/apache2/.htpasswd
                Require valid-user
        </Proxy>
</VirtualHost>
```

Verify our apache configuration:

```
sudo apachectl -t
```

you may get an error about the servers FQDN, that's not important.

Restart apache to load the changes:

```
sudo systemctl restart apache2
```

Configure Splunk to only accept connections from the local comptuer (redirected throug the proxy):

```
sudo vi /opt/splunk/etc/system/local/web.conf
```

Under the **settings** section, add one extra line: **server.socket_host = localhost** (if the file is empty, just add the following two lines, otherwise add server.socket to the settings section):

```
[settings]
server.socket_host = localhost
```

restart splunk to register the changes:

```
sudo service Splunkd restart
```

now try connecting to your splunk server on port 80, you should be prompted for a username and password. If you try connecting to port 8000, you should not be able to connect (unless you are connecting from the same computer).

Final Steps & System Cleanup

Congratulations, you should now have a complete Snort 3 NIDS with Splunk as your SIEM, and PulledPork to keep your rulesets updated. There are a few housekeeping items you may want to think about:

Remove Old Files: You may want to remove the snort_src directory, if you don't plan on compiling Snort again. You could also run **make clean** in all the sub-directories in there to reduce space instead.

Disable local Rules: you may want to disable the local rule(s) that were created, so they don't clog up your log file. Prepend a hash (#) to each line in that file to disable the rule. Reload Snort to detect the change.

Log Rotation: Snort will save logfiles into /var/log/snort. Neither Snort nor Splunk will delete those logs, so you'll need to look at some way to archive or delete the old logfiles (the .json logfiles from Snort IPS alerts, or the OpenAppID stats files, if you followed those optional instructions in Appendix A).

Adjust alert logged data: You may find that you don't need as much data from the Snort alerts as what's logged. You can remove some of the fields from the **alert_json** section in your snort.lua file. You may also want to adjust the size of the files that are written, depending on how often you plan on deleting the fies or how fast log files are generated.

You may want to install other Snort3 sensors on your network. If you do this, I recomend you install the Splunk Universal Forwarder (UF) rather than Splunk Enterprise on those additional systems, and configure that UF to send the log data back to your main Splunk server

Conclusion

See the Snort 3 blog for more information about running Snort 3 and compilation options. Snort 3 is much different from the Snort 2.9.9.x series. Both configuration and rule files are different, and not compatible between the two versions. Old Snort 2 configuration and rule files can be converted to the Snort 3 format using the included snort2lua command.

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

Appendix A: OpenAppID (Optional)

OpenAppID allows for the identification of application layer (layer 7) traffic. You can create rules that operate on application-layer traffic (for example to block facebook or a specifc type of VPN), and to log traffic statistics for each type of traffic detected. more info. OpenAppID is an optional feature that Snort offers, and you should enable it if you want to detect or block types of traffic (facebook, FTP, etc), or collect metrics on the ammount of data per type of traffic that is detected by your Snort server.

First let's enable the OpenAppID detectors (identifying traffic), then we'll enable the recording of OpenAppID metrics

The Snort team has put together a package of detectors with assistance from the community that you can download and install, called the **Application Detector Package**. First download the OpenAppID detector package and extract the files:

```
cd ~/snort_src/
wget https://snort.org/downloads/openappid/17843 -O OpenAppId-17843.tgz
tar -xzvf OpenAppId-17843.tgz
sudo cp -R odp /usr/local/lib/
```

Note: If you get an error that the file does not exist, it is possible that the Snort team updated the ruleset. Browse to https://snort.org/downloads#openappid, and download the **snort-openappid.tar.gz**.

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

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

Locate the following sections and configure as follows in the **Configure Inspection** section (arround line 90 or so in your snort.lua):

```
90  appid =
91  {
92      app_detector_dir = '/usr/local/lib',
93  }
```

Validate your snort.lua file as above since you've made changes.

modify our **local.rules** file with a new rule which will detect facebook traffic.

```
alert tcp any any -> any any ( msg:"Facebook Detected"; appids:"Facebook"; sid:10000002; metadata:policy security-ips alert; )
```

reload the snort3 service:

```
sudo service snort3 restart
```

generate some facebook traffic (`wget facebook.com`), and you'll see the alerts written to splunk (and to the json log file):

```
1  sourcetype="snort3:alert:json" msg="Facebook Detected"
```

Next, we can optionally configure Snort to capture OpenAppID statistics, basiclally how much data of each detected type is seen (how much DNS data, how much facebook data, how much https data, etc).

We need to download the **Snort Extras** repository, which holds additional inspectors and plugins, including the **appid_listener** which will allow us to output appid stats in JSON format.

```
cd ~/snort_src/
wget https://github.com/snort3/snort3_extra/archive/refs/tags/3.1.6.0.tar.gz -O snort3_extra-3.1.6.0.tar.gz
tar -xzvf snort3_extra-3.1.6.0.tar.gz
cd snort3_extra-3.1.6.0/
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig/
./configure_cmake.sh --prefix=/usr/local
cd build
make
sudo make install
```

This will install the snort_extras to /usr/local/lib/snort_extra.

Next we want to enable the appid_listener plugin from snort_extras. We do this by adding the following command to our snort.lua (a good place would be right after the appid section from above):

```
appid_listener =
{
    json_logging = true,
    file = "/var/log/snort/appid-output.log",
}
```

This is a good time to test that the snort_extras plugins can be loaded (that requires an additional command line opton), and that the new rule is correctly formatted. We run this with “sudo” because the appid_listener plugin tries to verify that it can open the logfile:

```
sudo snort -c /usr/local/etc/snort/snort.lua --plugin-path=/usr/local/lib/snort_extra
```

fix any errors you have before continuing. the **--plugin-path** paramter tells snort to load additional plugins from the extracted snort_extra package, including the **appid_listener** plugin.

Now we need to modfy our systemD Snort script to enabe it to load the appid_listener directory with the --plugin-path option:

```
sudo vi /lib/systemd/system/snort3.service
```

add the **--plugin-path** option:

```
ExecStart=/usr/local/bin/snort -c /usr/local/etc/snort/snort.lua -s 65535 \
    -k none -l /var/log/snort -D -u snort -g snort -i ens3 -m 0x1b --create-pidfile \
    --plugin-path=/usr/local/lib/snort_extra
```

NOTE: if you’re using PulledPork3 with the so_rules, you’ll need to move them from /usr/local/ect/so_rules to the snort_extra folder if you want to use both the snort_extras as well as the so_rules (you’ll also need to update the so_rue_path in your pulledpork.conf). This is because the plugin-path is used for both options, but can only point to one location.

reload the modified systemd file, reload snort, then verify that the service is running:

```
sudo systemctl enable snort3
sudo service snort3 restart
service snort3 status
```

Generate some network traffic (using wget, ping, or any other tool). You don’t have to match the rules, because we’re now collecting statistics for all the detectors. Check your log directory for **/var/log/snort/appid-output.log**, which contains these trafic statistics in json format:

```
{ "session_num": "0.58", "pkt_time": "2020-12-19 09:51:46.540562", "pkt_num": 1665, "apps": { "service"
    : "HTTPS", "client": "SSL client", "payload": "Facebook", "misc": null, "referred": null }, "proto"
    : "TCP", "client_info": { "ip": "10.10.10.88", "port": 33942, "version": null }, "service_info": {
    "ip": "185.60.216.35", "port": 443, "version": null, "vendor": null }, "user_info": { "id": 0, "
    username": null, "login_status": "n/a" }, "tls_host": "www.facebook.com", "dns_host": null, "http":
     { "http2_stream": null, "host": null, "url": null, "user_agent": null, "response_code": null, "
    referrer": null } }
```

The **service** field tells you which services were detected for this traffic flow (the ODF detector).

Next,we need to configure the Snort 3 JSON Alerts add-on to tell Splunk where the OpenAppID log files are stored. Just like before, we do this from the command line with a configuration file:

```
sudo vi /opt/splunk/etc/apps/TA_Snort3_json/local/inputs.conf
```

Add the following text into this **inputs.conf** file (don't remove the other section that points to the alerts file):

```
[monitor:///var/log/snort/*appid-output.log*]
sourcetype = snort3:openappid:json
```

restart Splunk:

```
sudo service Splunkd restart
```

now search splunk for OpenAppID data:

```
1 search sourcetype="snort3:openappid:json"
```