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 3.1.18.0.

Support: Read The Fine Manual (scroll down to **Resources**, the snort_reference manual has a lot of great information). If the manuals don't answer your question: you can 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 libjemalloc-dev
```

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 wget https://sourceforge.net/projects/pcre/files/pcre/8.45/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/1.77.0/source/boost_1_77_0.tar.gz
tar -xvzf boost_1_77_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_77_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/v3.0.5.tar.gz -O libdaq-3.0.5.tar.gz
tar -xzvf libdaq-3.0.5.tar.gz
cd libdaq-3.0.5
./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/3.1.18.0.tar.gz -O snort3-3.1.18.0.tar.gz
tar -xzvf snort3-3.1.18.0.tar.gz
cd snort3-3.1.18.0

./configure_cmake.sh --prefix=/usr/local --enable-tcmalloc --enable-jemalloc
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 3.1.18.0
   ''''    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 3.0.5
           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 170, 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):

```
171 ips =
172 {
173     enable_builtin_rules = true,
174     include = RULE_PATH .. "/local.rules",
175     variables = default_variables
176 }
```

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
```

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:

```
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**:

```
search_engine = { search_method = "hyperscan" }

detection = {
    hyperscan_literals = true,
    pcre_to_regex = true
}
```

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):

```
reputation =
{
   blocklist = BLACK_LIST_PATH .. "/default.blocklist",
}
```

Every time we modify our snort.lua we want to validate the file. If you're Using PulledPork3: you need to include the --plugin-path option pointing to the so_rules folder as well. Run one of the following two commands to test:

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

# or if you are using PulledPork3 with so_rules:
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:

```
230 alert_json =
231 {
232     file = true,
233     limit = 100,
234     fields = 'seconds action class b64_data dir dst_addr dst_ap dst_port eth_dst eth_len \
235     eth_src eth_type gid icmp_code icmp_id icmp_seq icmp_type iface ip_id ip_len msg mpls \
236     pkt_gen pkt_len pkt_num priority proto rev rule service sid src_addr src_ap src_port \
237     target tcp_ack tcp_flags tcp_len tcp_seq tcp_win tos ttl udp_len vlan timestamp',
238 }
```

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.
-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: