Snort 2.9.16.1 on Centos 8

Milad Rezaei

R&D, Shatel group of companies

October 13, 2020

Install

In this tutorial, we show how to install and configure snort 2.9.16.1 & Swatch & ELK stack on CentOS 8.

First, update the OS:

```
dnf update -y

dnf install epel-release -y
```

(We can install snort from source or install it using precompiled package exists in snort.org)

Snort provides rpm package for CentOS 8, which can be install simply with the below command:

```
dnf install https://www.snort.org/downloads/snort/

dnf install https://www.snort.org/downloads/snort/snort-2.9.16.1-1.centos8.x86_64.rpm
```

Installing from the source

Install necessary packages:

```
dnf install gcc gcc-c++ libnetfilter_queue-devel git flex bison zlib zlib-devel pcre
pcredevel libdnet tcpdump libnghttp2 wget xz-devel -y
```

We will download and store source files in the following folder:

```
mkdir ~/snort_src

cd ~/snort_src
```

Snort requires Libpcap and DAQ and we need to install them before installing snort:

Note:

Some network cards have features which can affect Snort. Two of these features are named "Large Receive Offload" (lro) and "Generic Receive Offload" (gro). With these features enabled, the network card performs packet reassembly before they're processed by the kernel.

By default, Snort will truncate packets larger than the default snaplen of 1518 bytes. In addition, LRO and GRO may cause issues with Stream target-based reassembly. We recommend that you turn off LRO and GRO. On linux systems, you can run:

```
ethtool -K eth1 gro off
ethtool -K eth1 lro off
```

Install libpcap

```
wget http://www.tcpdump.org/release/libpcap-1.8.1.tar.gz
tar xzvf libpcap-1.8.1.tar.gz
cd libpcap-1.8.1
./configure && make && make install
dnf install libpcap-devel -y
cd ..
```

Install DAQ

```
wget https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz
tar xvfz daq-2.0.7.tar.gz
cd daq-2.0.7
./configure && make && make install
cd ..
```

Install Snort:

```
wget https://www.snort.org/downloads/snort/snort-2.9.16.1.tar.gz
tar xvzf snort-2.9.16.1.tar.gz
cd snort-2.9.16.1
./configure --enable-sourcefire && make && make install
```

Configuration

Now we need to edit some configuration files, download the rules from snort.org and take snort for a test run.

First, we will update shared library:

```
ldconfig
```

Snort on CentOS is installed in /usr/local/bin/snort directory, it is a good practice to create a symbolic link to /usr/sbin/snort.

(If you installed Snort with ‘dnf’ you can skip this command.)

```
ln -s /usr/local/bin/snort /usr/sbin/snort
```

To verify the installation of snort use the command below:

```
snort -v
```

If you get error while loading shared libdnet.1 libraries, create the following link and try again.

```
ln -s /usr/lib64/libdnet.so.1.0.1 /usr/lib64/libdnet.1
```

To run Snort on CentOS safely without root access, we should create a new unprivileged user and a new user group for the daemon.

(If you installed Snort with ‘dnf’ you can skip this command.)

```
groupadd snort

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

Then create the folder structure to keep the Snort configuration, use commands below. If you installed Snort using dnf these directories should have already been added at install, check to make sure.

```
mkdir -p /etc/snort/rules

mkdir /var/log/snort

mkdir /usr/local/lib/snort_dynamicrules
```

Set the permissions for the new directories:

```
chmod -R 5775 /etc/snort

chmod -R 5775 /var/log/snort

chmod -R 5775 /usr/local/lib/snort_dynamicrules

chown -R snort:snort /var/log/snort

chown -R snort:snort /usr/local/lib/snort_dynamicrules
```

Create new files:

```
touch /etc/snort/rules/white_list.rules

touch /etc/snort/rules/black_list.rules

touch /etc/snort/rules/local.rules
```

The black_list and white_list are used for Reputation preprocessor. You can use these 2 files if you want to write your lists for Reputation preprocessor or use this file /etc/snort/rules/iplists/default.blocklist . The second file will get form snort by Pulled pork.

Pulledpork

Pulled_Pork is tool written in perl for managing Snort rule sets. Pulled_Pork features include:

Automatic rule downloads using your Oinkcode

MD5 verification prior to downloading new rulesets

Full handling of Shared Object (SO) rules

Generation of so_rule stub files

Modification of ruleset state (disabling rules, etc)

The project is run by JJ Cummings

Install necessary packages:

```
dnf install perl-libwww-perl perl-core  perl-LWP-Protocol-https
```

Download Pulledpork from Git and install:

```
git clone https://github.com/shirkdog/pulledpork.git
cd pulledpork/
cp pulledpork.pl /usr/local/bin
chmod +x /usr/local/bin/pulledpork.pl
cp etc/*.conf /etc/snort
mkdir /etc/snort/rules/iplists
touch /etc/snort/rules/iplists/default.blocklist
```

To verify the installation of Pulledpork use the command below:

```
pulledpork.pl -V
```

Run these commands to change rules path on snort.conf and make some files:

```
echo "include \$RULE_PATH/so_rules.rules" >> /etc/snort/snort.conf
echo "include \$RULE_PATH/snort.rules" >> /etc/snort/snort.conf
touch /etc/snort/rules/so_rules.rules
touch /etc/snort/rules/snort.rules
```

Then make change to Pulledpork config file like below: replace your oinkcode

```
vi /etc/snort/pulledpork.conf
rule_url=https://www.snort.org/reg-rules/|snortrules-snapshot.tar.gz| <oinkcode>
line 72 change to:rule_path=/etc/snort/rules/snort.rules
line 87 change to:local_rules=/etc/snort/rules/local.rules
line 90 change to:sid_msg=/etc/snort/sid-msg.map
line 117 change to:config_path=/etc/snort/snort.conf
line 134 change to:distro=Centos-7
line 142 change to:block_list=/etc/snort/rules/iplists/default.blocklist
line 151 change to:IPRVersion=/etc/snort/rules/iplists
line 200 uncomment and change to:enablesid=/etc/snort/enablesid.conf
line 201 uncomment and change to:dropsid=/etc/snort/dropsid.conf
line 202 uncomment and change to:disablesid=/etc/snort/disablesid.conf
line 203 uncomment and change to:modifysid=/etc/snort/modifysid.conf
save and quite
```

Runing Pulledpork:

```
pulledpork.pl -c /etc/snort/pulledpork.conf
```

If you get (*The specified Snort binary does not exist!*

Please correct the value or specify the FULL rules tarball name in the pulledpork.conf!

at /usr/local/bin/pulledpork.pl line 2120.) error do like this :

```
Vi /etc/snort/pulledpork
Line 113 : snort_path=/usr/sbin/snort
```

To make Pulledpork run automatically please visit https://snort.org/oinkcodes and read instruction.
For example: (make sure use their command)

```
crontab –e

17 15 * * * root /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf
```

Configuring the network and rule sets

Edit snort.conf file to modify a few parameters:

```
vi /etc/snort/snort.conf
```

Change parameter as the example below:

```
ipvar HOME_NET 192.168.1.0/24

ipvar EXTERNAL_NET !$HOME_NET

var RULE_PATH /etc/snort/rules

var SO_RULE_PATH /etc/snort/so_rules

var PREPROC_RULE_PATH /etc/snort/preproc_rules

var WHITE_LIST_PATH /etc/snort/rules

var BLACK_LIST_PATH /etc/snort/rules
```

In this tutorial, we use ELK stack to store and visualize alerts and log from snort log folder . For that reason, we need to setup snort output log as below:

```
# syslog

 output alert_syslog: LOG_LOCAL2 LOG_ALERT
```

Finally test snort configuration file by the following command:

```
snort -T -c /etc/snort/snort.conf
```

If you get success message everything is correct.

To test Snort we add rules to local. Rules:

```
vi /etc/snort/rules/local.rules

alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)
```

These rules make alerts for all icmp messages being sent to $HOME_NET (snort alert structure could be found in snort.org)

Selinux

Disable selinux or set it to Permissive mode

Running snort as a Daemon

(If you installed Snort with ‘dnf’ you can skip this section.)

To run snort on CentOS as a service in the background you should copy following script to /etc/init.d/

```
vi /etc/init.d/snortd
```

```
#!/bin/sh

# $Id$

#

# snortd        Start/Stop the snort IDS daemon.

#

# chkconfig: 2345 40 60

# description:  snort is a lightweight network intrusion detection tool that \

#               currently detects more than 1100 host and network \

#               vulnerabilities, portscans, backdoors, and more.

#


# Source function library.

. /etc/rc.d/init.d/functions


# Source the local configuration file

. /etc/sysconfig/snort


# Convert the /etc/sysconfig/snort settings to something snort can

# use on the startup line.

if [ "$ALERTMODE"X = "X" ]; then

   ALERTMODE=""

else

   ALERTMODE="-A $ALERTMODE"

fi


if [ "$USER"X = "X" ]; then

   USER="snort"

fi
```

```
if [ "$GROUP"X = "X" ]; then

  GROUP="snort"

fi


if [ "$BINARY_LOG"X = "1X" ]; then

  BINARY_LOG="-b"

else

  BINARY_LOG=""

fi


if [ "$CONF"X = "X" ]; then

  CONF="-c /etc/snort/snort.conf"

else

  CONF="-c $CONF"

fi


if [ "$INTERFACE"X = "X" ]; then

  INTERFACE="-i ens33"

else

  INTERFACE="-i $INTERFACE"

fi


if [ "$DUMP_APP"X = "1X" ]; then

  DUMP_APP="-d"

else

  DUMP_APP=""

fi


if [ "$NO_PACKET_LOG"X = "1X" ]; then

  NO_PACKET_LOG="-N"

else

  NO_PACKET_LOG=""

fi
```

```
if [ "$PRINT_INTERFACE"X = "1X" ]; then
  PRINT_INTERFACE="-I"
else
  PRINT_INTERFACE=""
fi


if [ "$PASS_FIRST"X = "1X" ]; then
  PASS_FIRST="-o"
else
  PASS_FIRST=""
fi


if [ "$LOGDIR"X = "X" ]; then
  LOGDIR=/var/log/snort
fi


# These are used by the 'stats' option
if [ "$SYSLOG"X = "X" ]; then
  SYSLOG=/var/log/messages
fi


if [ "$SECS"X = "X" ]; then
  SECS=5
Fi
if [ ! "$BPFFILE"X = "X" ]; then
  BPFFILE="-F $BPFFILE"
fi


######################################
# Now to the real heart of the matter:
# See how we were called.
case "$1" in
 start)
      echo -n "Starting snort: "
      cd $LOGDIR
```

```
if [ "$INTERFACE" = "-i ALL" ]; then
        for i in `cat /proc/net/dev|grep eth|awk -F ":" '{ print $1; }'`
        do
            mkdir -p "$LOGDIR/$i"
            chown -R $USER:$GROUP $LOGDIR
            daemon /usr/sbin/snort $ALERTMODE $BINARY_LOG $NO_PACKET_LOG $DUMP_APP -D $PRINT_INTERFACE -i
$i -u $USER -g $GROUP $CONF -l $LOGDIR/$i $PASS_FIRST $BPFFILE $BPF
        done
      else
        # check if more than one interface is given
        if [ `echo $INTERFACE|wc -w` -gt 2 ]; then
          for i in `echo $INTERFACE | sed s/"-i "//`
            do
              mkdir -p "$LOGDIR/$i"
              chown -R $USER:$GROUP $LOGDIR
              daemon /usr/sbin/snort $ALERTMODE $BINARY_LOG $NO_PACKET_LOG $DUMP_APP -D $PRINT_INTERFACE -i
$i -u $USER -g $GROUP $CONF -l $LOGDIR/$i $PASS_FIRST $BPFFILE $BPF
          done
        else
          # Run with a single interface (default)
          daemon /usr/sbin/snort $ALERTMODE $BINARY_LOG $NO_PACKET_LOG $DUMP_APP -D $PRINT_INTERFACE
$INTERFACE -u $USER -g $GROUP $CONF -l $LOGDIR $PASS_FIRST $BPFFILE $BPF
        fi
      fi
      touch /var/lock/subsys/snort
      echo
      ;;
  stop)
echo -n "Stopping snort: "
      killproc snort
      rm -f /var/lock/subsys/snort
      echo
      ;;
  reload)
echo "Sorry, not implemented yet"
      ;;
```

```
restart)
     $0 stop
     $0 start
     ;;
 condrestart)
     [ -e /var/lock/subsys/snort ] && $0 restart
     ;;
 status)
     status snort
     ;;
 stats)
     TC=125                    # Trailing context to grep
     SNORTNAME='snort'          # Process name to look for

     if [ ! -x "/sbin/pidof" ]; then
       echo "/sbin/pidof not present, sorry, I cannot go on like this!"
       exit 1
     fi

     #Grab Snort's PID
     PID=`pidof -o $$ -o $PPID -o %PPID -x ${SNORTNAME}`

     if [ ! -n "$PID" ]; then       # if we got no PID then:
       echo "No PID found: ${SNORTNAME} must not running."
       exit 2
     fi

     echo ""
     echo "*******"
     echo "WARNING:  This feature is EXPERIMENTAL - please report errors!"
     echo "*******"
     echo ""
     echo "You can also run: $0 stats [long | opt]"
     echo ""
     echo "Dumping ${SNORTNAME}'s ($PID) statistics"
     echo "please wait..."
```

```
# Get the date and tell Snort to dump stats as close together in
    # time as possible--not 100%, but it seems to work.
    startdate=`date '+%b %e %H:%M:%S'`

    # This causes the stats to be dumped to syslog
    kill -USR1 $PID

    # Sleep for $SECS secs to give syslog a chance to catch up
    # May need to be adjusted for slow/busy systems
    sleep $SECS

    if [ "$2" = "long" ]; then          # Long format
      egrep -B 3 -A $TC "^$startdate .* snort.*: ={79}" $SYSLOG | \
        grep snort.*:
    elif [ "$2" = "opt" ]; then         # OPTimize format
     # Just show stuff useful for optimizing Snort
      egrep -B 3 -A $TC "^$startdate .* snort.*: ={79}" $SYSLOG | \
        egrep "snort.*: Snort analyzed |snort.*: dropping|emory .aults:"
    else                                # Default format
      egrep -B 3 -A $TC "^$startdate .* snort.*: ={79}" $SYSLOG | \
        grep snort.*: | cut -d: -f4-
    fi
    ;;
 *)
    echo "Usage: $0 {start|stop|reload|restart|condrestart|status|stats (long|opt)}"
    exit 2
esac
exit 0
```

Then:

```
vi /etc/sysconfig/snort
```

And copy this script to that file (replace interface name) :

```
# /etc/sysconfig/snort

# $Id$

# All of these options with the exception of -c, which tells Snort where

# the configuration file is, may be specified in that configuration file as

# well as the command line. Both the command line and config file options

# are listed here for reference.

#### General Configuration


# What interface should snort listen on?  [Pick only 1 of the next 3!]

# This is -i {interface} on the command line

# This is the snort.conf config interface: {interface} directive

INTERFACE=ens33

#

# The following two options are not directly supported on the command line

# or in the conf file and assume the same Snort configuration for all

# instances

#

# To listen on all interfaces use this:

#INTERFACE=ALL

#

# To listen only on given interfaces use this:

#INTERFACE="eth1 eth2 eth3 eth4 eth5"

# Where is Snort's configuration file?

# -c {/path/to/snort.conf}

CONF=/etc/snort/snort.conf
```

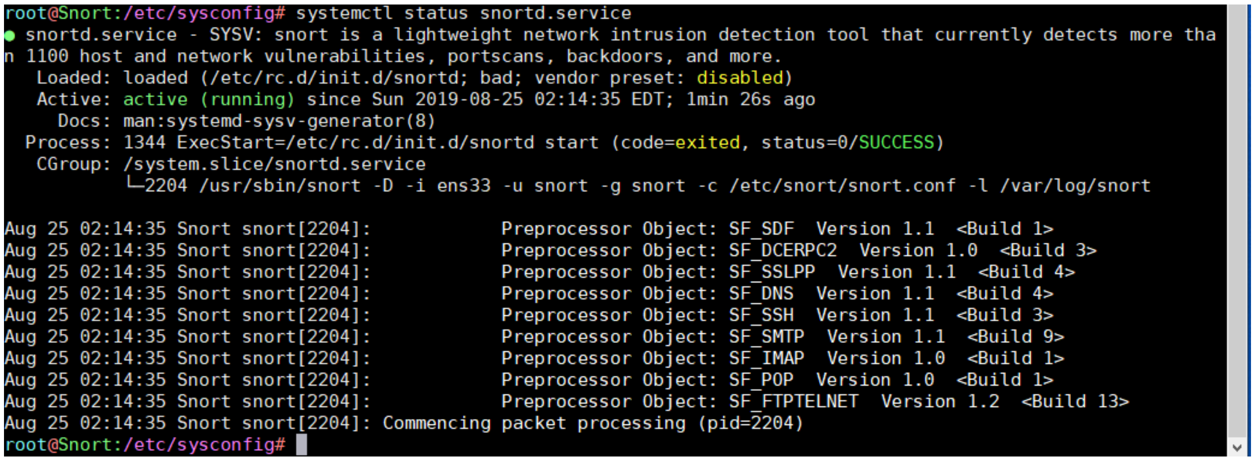
```
# What user and group should Snort drop to after starting? This user and

# group should have very few privileges.

# -u {user} -g {group}

# config set_uid: user

# config set_gid: group

USER=snort

GROUP=snort


# Should Snort change the order in which the rules are applied to packets.

# Instead of being applied in the standard Alert->Pass->Log order, this will

# apply them in Pass->Alert->Log order.

# -o

# config order: {actions in order}

# e.g. config order: log alert pass activation dynamic suspicious redalert

PASS_FIRST=0



#### Logging & Alerting


# NOTE: NO_PACKET_LOG and BINARY_LOG, ALERTMODE, etc. are mutually

# exclusive. Use either NO_PACKET_LOG or any/all of the other logging

# options. But the more logging options use you, the slower Snort will run.

# Where should Snort log?

# -l {/path/to/logdir}

# config logdir: {/path/to/logdir}

LOGDIR=/var/log/snort

# How should Snort alert? Valid alert modes include fast, full, none, and

# unsock.  Fast writes alerts to the default "alert" file in a single-line,

# syslog style alert message.  Full writes the alert to the "alert" file

# with the full decoded header as well as the alert message.  None turns off

# alerting. Unsock is an experimental mode that sends the alert information

# out over a UNIX socket to another process that attaches to that socket.

# -A {alert-mode}

# output alert_{type}: {options}

#ALERTMODE=full
```

```
# Should Snort dump the application layer data when displaying packets in
# verbose or packet logging mode.
# -d
# config dump_payload
#DUMP_APP=1

# Should Snort keep binary (AKA pcap, AKA tcpdump) logs also? This is
# recommended as it provides very useful information for investigations.
# -b
# output log_tcpdump: {log name}
#BINARY_LOG=0

# Should Snort turn off packet logging?  The program still generates
# alerts normally.
# -N
# config nolog
NO_PACKET_LOG=0

# Print out the receiving interface name in alerts.
# -I
# config alert_with_interface_name
PRINT_INTERFACE=0
# When dumping the stats, what log file should we look in
SYSLOG=/var/log/messages

# When dumping the stats, how long to wait to make sure that syslog can
# flush data to disk
SECS=5
# To add a BPF filter to the command line uncomment the following variable
# syntax corresponds to tcpdump(8)
#BPF="not host 192.168.1.1"
# To use an external BPF filter file uncomment the following variable
# syntax corresponds to tcpdump(8)
# -F {/path/to/bpf_file}
# config bpf_file: /path/to/bpf_file
#BPFFILE=/etc/snort/bpf_file
```

If you install Snort using dnf, you should already have the startup script configured. Start the service as described below.

```
chmod 755  /etc/init.d/snortd

systemctl daemon-reload

systemctl start snortd

systemctl enable snortd
```

If we use systemctl status snortd.service we should see output like below:



```
root@Snort:/etc/sysconfig# systemctl status snortd.service
● snortd.service - SYSV: snort is a lightweight network intrusion detection tool that currently detects more tha
n 1100 host and network vulnerabilities, portscans, backdoors, and more.
   Loaded: loaded (/etc/rc.d/init.d/snortd; bad; vendor preset: disabled)
   Active: active (running) since Sun 2019-08-25 02:14:35 EDT; 1min 26s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1344 ExecStart=/etc/rc.d/init.d/snortd start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/snortd.service
           └─2204 /usr/sbin/snort -D -i ens33 -u snort -g snort -c /etc/snort/snort.conf -l /var/log/snort

Aug 25 02:14:35 Snort snort[2204]:           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
Aug 25 02:14:35 Snort snort[2204]:           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Aug 25 02:14:35 Snort snort[2204]:           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
Aug 25 02:14:35 Snort snort[2204]:           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
Aug 25 02:14:35 Snort snort[2204]:           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
Aug 25 02:14:35 Snort snort[2204]:           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
Aug 25 02:14:35 Snort snort[2204]:           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
Aug 25 02:14:35 Snort snort[2204]:           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
Aug 25 02:14:35 Snort snort[2204]:           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
Aug 25 02:14:35 Snort snort[2204]: Commencing packet processing (pid=2204)
root@Snort:/etc/sysconfig# 
```

In /etc/sysconfig/snort file we can determine how snort starts and sniffs on which interface or determine how to save output logs. comment some variables in this file like:

```
vi /etc/sysconfig/snort
```

Comment this variable:

```
BINARY_LOG=0

DUMP_APP=1

ALERTMODE=full
```

Save and exit. Now, snort has been installed and ready to use in Nids mode.

Sending alert by email

To send alert by email we need to configure snort to send log to rsyslog:

```
Vi /etc/snort/snort.conf

Line 528 uncommet : output alert_syslog: LOG_LOCAL2 LOG_ALERT
```

Then configure rsyslog to store received log from snort to /var/log/alert.log:

```
vi /etc/rsyslog.conf

add this line

local2.alert                              /var/log/snort/alert.log
```

Now install swatch

```
dnf install swatch
```

now create folder for swatch config file

```
mkdir ~/swatch

vi se.conf
```

now we need to configure swatch to find specific word on taild log file

in this case I configured swatch to find alert that contain “Priority: 1” & “Priority: 2”

```
watchfor  /[Priority: (1|2)]/

      echo red

      mail=root@localhost, subject="Nids : Priority: $1"
```

for run as daemon and using new config file

for run in stratup make swatch.sh and copy upper command into it and append :

```
Vi /etc/rc.local
Sh /root/swatch/swatch.sh
```

send alert by snort-watcher

if you use base , snort-watcher can looking base database to find new alert and send them

you can find it from Github :

https://github.com/miladstar77/Snort-Watcher

ELK Stack

Instead of base and banyard2 we can use ELK stack to store and visualize alerts .

Elasticsearch is an open source search engine based on Lucene, developed in Java. It provides a distributed and multitenant full-text search engine with an HTTP Dashboard web-interface (Kibana). The data is queried, retrieved and stored with a JSON document scheme. Elasticsearch is a scalable search engine that can be used to search for all kind of text documents, including log files. Elasticsearch is the heart of the 'Elastic Stack' or ELK Stack.

Logstash is an open source tool for managing events and logs. It provides real-time pipelining for data collections. Logstash will collect your log data, convert the data into JSON documents, and store them in Elasticsearch.

Kibana is an open source data visualization tool for Elasticsearch. Kibana provides a pretty dashboard web interface. It allows you to manage and visualize data from Elasticsearch. It's not just beautiful, but also powerful.

We must install ELK on another system or if we want to install ELK on snort system we should have 8GB Ram .

first install java

```
dnf install java-1.8.0-openjdk

java -version
```

Example out;

```
java -version
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

In this step, we will install and configure Elasticsearch. we will install Elasticsearch from an rpm package provided by elastic.co and configure it to run on localhost (to make the setup secure and ensure that it is not reachable from the outside).

```
Wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.9.2-x86_64.rpm

rpm -i elasticsearch-7.9.2-x86_64.rpm
```

configure elasticsearch :

Enable memory lock for Elasticsearch . This disables memory swapping for Elasticsearch.

```
vi /etc/elasticsearch/elasticsearch.conf

uncomment line 43 : bootstrap.memory_lock: true

line 55 :  network.host: "localhost"

line 59 : http.port: 9200

vi /etc/sysconfig/elasticsearch

line 46 MAX_LOCKED_MEMORY=unlimited
```

```
systemctl daemon-reload
systemctl enable elasticsearch
systemctl start elasticsearch
```

check that Elasticsearch is running:

```
curl -XGET 'localhost:9200/?pretty'
```

install and configure Kibana :

we will install and configure Kibana with a Nginx web server. Kibana will listen on the localhost IP address and Nginx acts as a reverse proxy for the Kibana application.

```
Wget https://artifacts.elastic.co/downloads/kibana/kibana-7.9.2-x86_64.rpm

rpm –i kibana-7.9.2-x86_64.rpm
```

Configure Kibana :

```
vi /etc/kibana/kibana.yml

server.port: 5601
server.host: "localhost"
elasticsearch.url: "http://localhost:9200"
```

```
systemctl enable kibana
systemctl start kibana
```

install the Nginx and httpd-tools package:

```
dnf -y install nginx httpd-tools
```

configure nginx :

```
vi /etc/nginx/nginx.conf

<<<Remove the server { } block>>>
```

then create user and password for access to Kibana :

```
htpasswd –c /etc/nginx/htpasswd.kibana admin
```

Now we need to create a new virtual host configuration file in the conf.d directory. Create the new file 'kibana.conf' with vi

```
vi /etc/nginx/conf.d/kibana.conf

server {
    listen 80;

    server_name snort;

    auth_basic "Restricted Access";
    auth_basic_user_file /etc/nginx/htpasswd.kibana;

    location / {
        proxy_pass http://localhost:5601;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Test and run nginx:

```
nginx -t
systemctl enable nginx
systemctl start nginx
```

Install and Configure Logstash

```
Wget https://artifacts.elastic.co/downloads/logstash/logstash-7.9.2.rpm

rpm –i  logstash-7.9.2.rpm
```

Configure Logstash :

by Logstash we collect logs then filter them. we receive logs from filebeat :

```
vi /etc/logstash/conf.d/10_input_beats.logstash.conf

input {

  beats {

    port => 5044

  }

}
```

```
vi /etc/logstash/conf.d/20_filter.conf

filter {

      grok {

                match => { "message" => "%{SYSLOGTIMESTAMP:timestamp}
%{HOSTNAME:sensor_name} snort\[%{NUMBER:GID}\]\: \[%{DATA:SIG_ID}\]
%{DATA:MSG} \[Classification\: %{DATA:Calssification}\] \[Priority\:
%{NUMBER:Priority}\] \{%{WORD:Protocol}\} %{IP:Src_IP}\:%{INT:Src_Port} \-\>
%{IP:Dst_IP}\:%{INT:Dst_Port}"}

      }

}
```

```
vi /etc/logstash/conf.d/30_output_elasticsearch.logstash.conf

output {

  elasticsearch {

    hosts => [ "127.0.0.1:9200" ]

    index => "snort-%{+YYYY.MM.dd}"

  }

}
```

run logstash :

```
systemctl enable logstash
systemctl start logstash
```

Install and configure Filebeat:

We can send our snort alert from log file by filebeat to logstash .

If you have more than 1 sensor you must install filebeat on each sensor and send alert to main system.

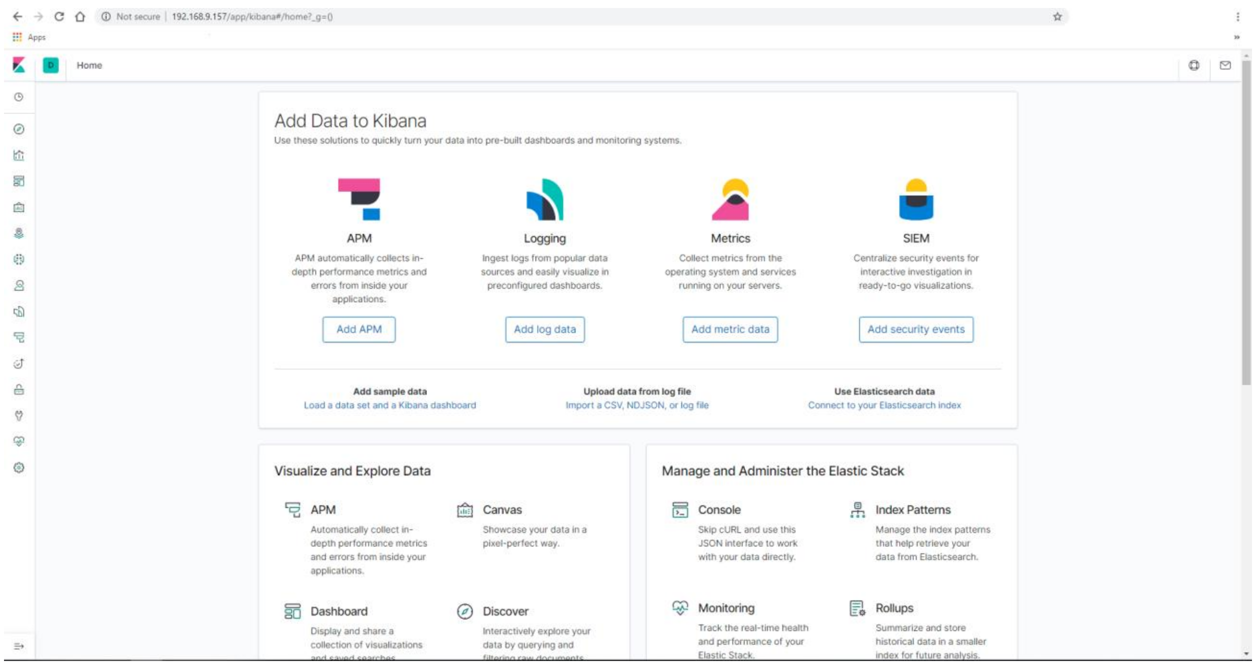
```
Wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.9.2-x86_64.rpm

rpm –i filebeat-7.9.2-x86_64.rpm
```

```
vi /etc/filebeat/filebeat.yml

line 24 : enabled: true

line 27 : paths:

                          - /var/log/snort/alert.log

                          document_type: snort

line 159 : output.logstash:

                                                  # The Logstash hosts

                                                   hosts: ["localhost:5044"]

comment line 149 and 151
```

if you want to send logs to another system change ["localhost:5044"] to ["mainipaddress:5044"] run filebeat:

```
systemctl enable filebeat

systemctl start filebeat
```

Open your web browser and insert your ip address to see kibana:


Not secure | 192.168.9.157/app/kibana#/home?_g=()
Apps
Home
Add Data to Kibana
APM
Logging
Metrics
SIEM
Add APM
Add log data
Add metric data
Add security events
Add sample data
Load a data set and a Kibana dashboard
Upload data from log file
Import a CSV, NDJSON, or log file
Use Elasticsearch data
Connect to your Elasticsearch index
Visualize and Explore Data
Manage and Administer the Elastic Stack
APM
Canvas
Dashboard
Discover
Console
Index Patterns
Monitoring
Rollups

Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.

APM automatically collects in-depth performance metrics and errors from inside your applications.

Ingest logs from popular data sources and easily visualize in preconfigured dashboards.

Collect metrics from the operating system and services running on your servers.

Centralize security events for interactive investigation in ready-to-go visualizations.

Automatically collect in-depth performance metrics and errors from inside your applications.

Showcase your data in a pixel-perfect way.

Display and share a collection of visualizations and saved searches.

Interactively explore your data by querying and filtering raw documents.

Skip cURL and use this JSON interface to work with your data directly.

Manage the index patterns that help retrieve your data from Elasticsearch.

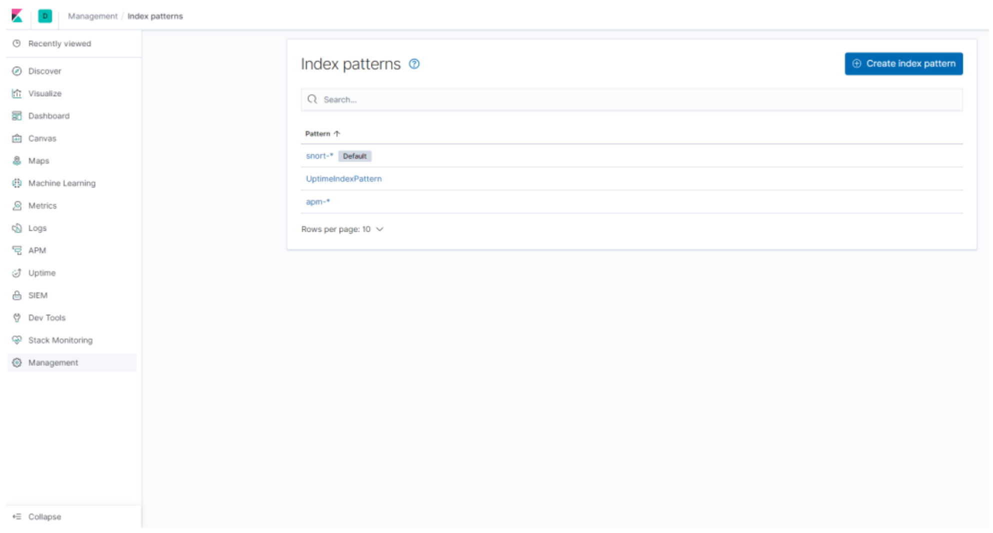
Track the real-time health and performance of your Elastic Stack.

Summarize and store historical data in a smaller index for future analysis.

To add snort index to Kibana from menu :

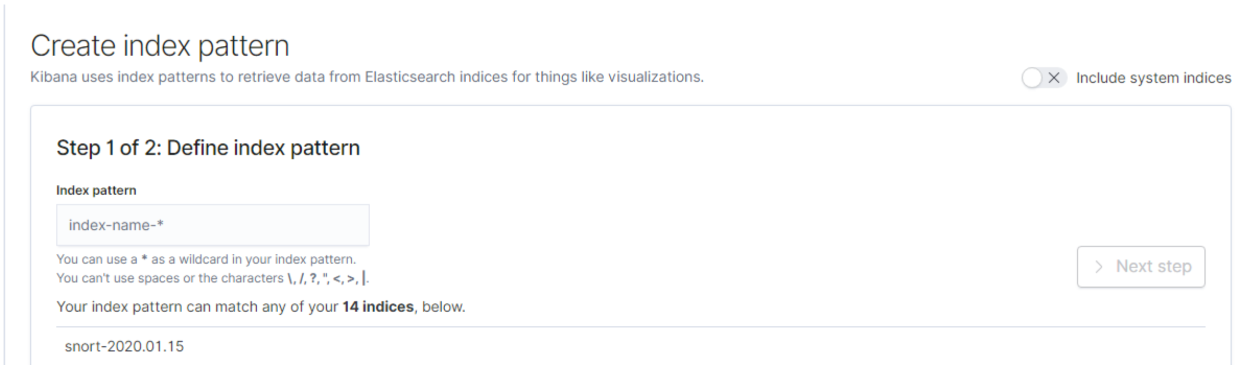
Open Management/Index patterns

Then : Create index pattern



In index pattern search for : snort-*

Then add index to kibana . if you didn’t see index go back and check installation of your stack.



Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Include system indices

Step 1 of 2: Define index pattern

Index pattern

index-name-*

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

Next step

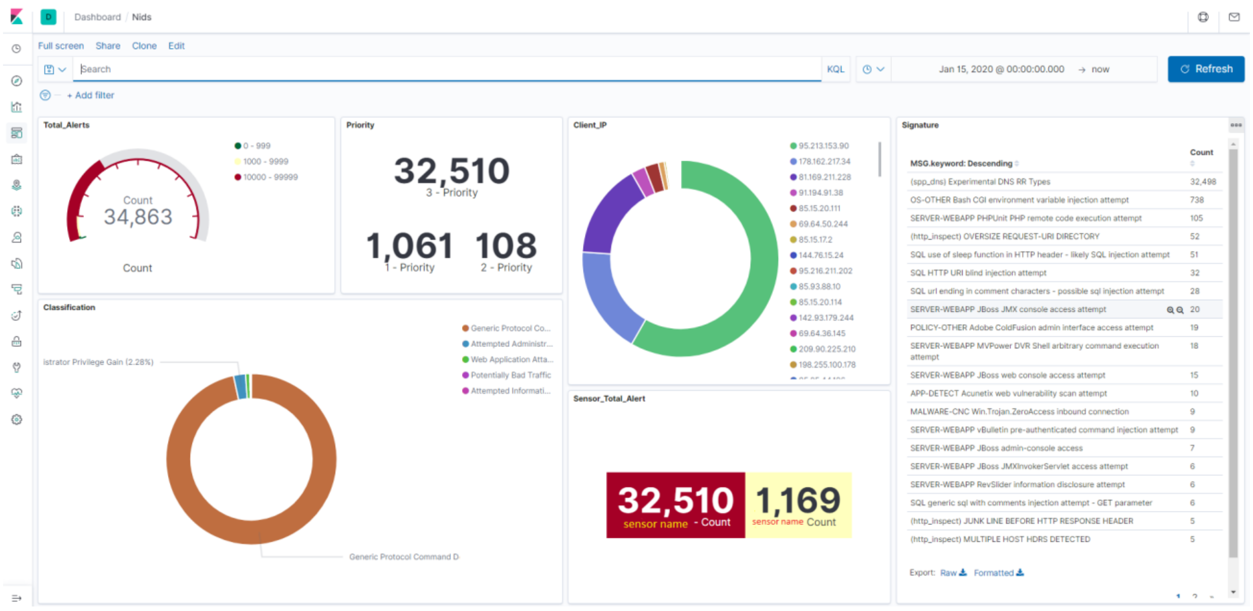
Your index pattern can match any of your **14 indices**, below.

snort-2020.01.15

After index added to Kibana you can see your logs .

You can create graph in visualizations menu and create dashboard .

This is my dashboard . if you have any question you can send email to cod.smr@gmail.com.



MSG.keyword: Descending	Count
(spp_dns) Experimental DNS RR Types	32,498
OS-OTHER Bash CGI environment variable injection attempt	738
SERVER-WEBAPP PHPUnit PHP remote code execution attempt	105
(http_inspect) OVERSIZE REQUEST-URI DIRECTORY	52
SQL use of sleep function in HTTP header - likely SQL injection attempt	51
SQL HTTP URI blind injection attempt	32
SQL url ending in comment characters - possible sql injection attempt	28
SERVER-WEBAPP JBoss JMX console access attempt	20
POLICY-OTHER Adobe ColdFusion admin interface access attempt	19
SERVER-WEBAPP MVPower DVR Shell arbitrary command execution attempt	18
SERVER-WEBAPP JBoss web console access attempt	15
APP-DETECT Acunetix web vulnerability scan attempt	10
MALWARE-CNC Win.Trojan.ZeroAccess inbound connection	9
SERVER-WEBAPP vBulletin pre-authenticated command injection attempt	9
SERVER-WEBAPP JBoss admin-console access	7
SERVER-WEBAPP JBoss JMXInvokerServlet access attempt	6
SERVER-WEBAPP RevSlider information disclosure attempt	6
SQL generic sql with comments injection attempt - GET parameter	6
(http_inspect) JUNK LINE BEFORE HTTP RESPONSE HEADER	5
(http_inspect) MULTIPLE HOST HDRS DETECTED	5