INSTALLATION OF SNORT-2.9.17 AND BASE-1.4.5 ON UBUNTU 16.04

VERSION 1.0

DATE : 24 MARCH 2021

PERFORMED BY

VIGNESHWAR SETHURAMAN

TABLE OF CONTENTS

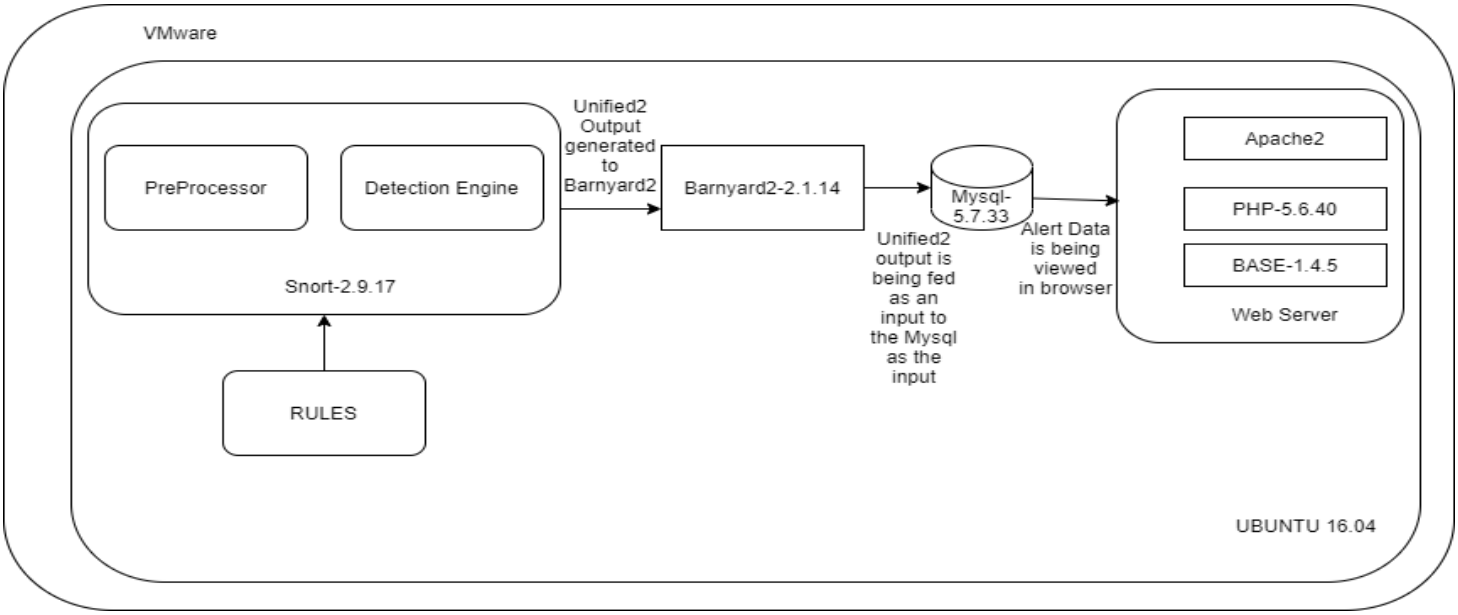
LIST OF FIGURES

PART I:INTRODUCTION

In this document, the steps that detail the installation of Snort as a NIDS (network intrusion detection system), with three pieces of additional software to improve the functionality of Snort, are given. This guide is written with the Snort host as a VMware workstation pro virtual machine but can easily install Snort on a physical machine or as a virtual machine on another platform. This installer document has been tested on the Ubuntu 16.04 Desktop x64.
While the Snort can be installed without any supporting software, it will work just fine, and it becomes much more helpful with a few additional software packages.
These packages are Barnyard2:
Software that takes Snort output and writes to a SQL database, which reduces the load on the system.
BASE:
A web-based graphical interface for viewing and clearing Snort events.
Note: while this document focuses on the current 2.9.x series release of Snort, these steps will most likely work to install the older Snort 2.9.8.x series, and could be used to install Snort on older or derivative versions of Ubuntu (Xubuntu, Mint, etc.)
Software versions used in this guide:
• Snort 2.9.17
• Barnyard2 2-1.14 (current master)
• BASE 1.4.5

Administrator Accounts: This document assumes that the user is logged into the system as a regular user and will run all administrative commands with sudo. This helps to identify what commands require administrative credentials and which do not. We will also create a non-privileged user named snort that will have access to run all applications when setting up services.

PART II:TOPOLOGY



PART III:INSTALLATION

INSTALLING UBUNTU

This document will assume that there is already an installed version of Ubuntu with all the default settings. Snort does not need an IP address assigned to the interface that it is listening on, and in many configurations, snort will listen on

interfaces that do not have an IP address configured. For this document, it is easier to manage the system remotely via ssh if the interface is reachable. It is recommended to have one interface on the Snort server for management in a production environment and have Snort listen on other interfaces, but this is not required. By default, Ubuntu will use DHCP to auto-configure an address. If this is the case, it can be verified by checking the ip address by running ifconfig eth0. If there is no DHCP server assigned, then the IP addresses must be configured on the Snort system manually. There should be internet connectivity to download the required packages and software tarballs. Once the user is logged in for the first time and verified the internet connectivity, then also must verify the system is up to date and install openssh-server (As to remotely-manage the system).

Reboot after installation to make sure all patches are applied.

 # INSTALL UPDATES AND REBOOT:

sudo apt-get update
sudo apt-get dist-upgrade -y
sudo apt-get install -y openssh-server
sudo reboot

INSTALLING THE SNORT PRE-REQUISITES

Snort has four main pre-requisites:

- pcap (libpcap-dev) available from the Ubuntu repository
- PCRE (libpcre3-dev) available from the Ubuntu repository
- Libdnet (libdumbnet-dev) available from the Ubuntu repository
- DAQ (http://www.snort.org/downloads/) compiled from source

Initially, there should be an install for all the tools happen that are required for building software. The build essentials package does this:

sudo apt-get install -y build-essential

Once the build tools are installed, then the install for all Snort pre-requisites that are available from the Ubuntu repositories3 can be performed:

sudo apt-get install -y libpcap-dev libpcre3-dev libdumbnet-dev

The Snort DAQ (Data Acquisition library)has a few pre-requisites that need to be installed:

sudo apt-get install -y bison flex

In this document, we will be downloading several tarballs for various software packages. We will create a folder called snort src to keep them all in one place:

mkdir ~/snort_src
cd ~/snort_src

Download and install the latest version of DAQ from the Snort website. The steps below use wget to download version 2.0.7 of DAQ, which is the newest version at the time of writing this document.

cd ~/snort_src
wget https://snort.org/downloads/snort/daq-2.0.7.tar.gz
tar -xvzf daq-2.0.7.tar.gz
cd daq-2.0.7
./configure
make
sudo make install

INSTALLING SNORT

To install Snort on Ubuntu, one additional required pre-requisite needs to be installed that is not mentioned in the documentation: zlibg, a compression library. Four optional libraries improve functionality: liblzma-dev, three of which provide decompression of swf files (adobe ash), openssl, and libssl-dev which both provide SHA and MD5 file signatures:

sudo apt-get install -y zlib1g-dev liblzma-dev openssl libssl-dev

We need the development libraries for Nghttp2: an HTTP/2 C Library which implements the HPAC header compression algorithm. In Ubuntu 16, the install is easy:

 # UBUNTU 16 ONLY

sudo apt-get install -y libnghttp2-dev

cd ~/snort_src
wget https://snort.org/downloads/snort/snort-2.9.17.tar.gz
tar -xvzf snort-2.9.17.tar.gz
cd snort-2.9.17
./configure --enable-sourcefire
make
sudo make install

Note: If there occurs any configure error: "Fatal!" when running ./configure, it seems fine to continue. If there arises an error, then it should be resolved before continuing. The output from ./configure is piped into grep "... no" to get a list of all software that did not install. ./configure is ran more than once, first to make sure that there are no overall issues, then again to see what optional components did not install: ./configure | grep "... no"

Run the following command to update shared libraries (you will get an error when you try to run Snort if you skip this step):
sudo ldconfig

Place a symlink to the Snort binary in /usr/sbin:
sudo ln -s /usr/local/bin/snort /usr/sbin/snort
Test Snort by running the binary as a regular user, passing it the -V flag (which tells Snort to verify itself and any configuration files passed to it). You should see output like what is shown below (although exact version numbers may be slightly different):

```
root@ubuntu:/home/vignesh/Downloads# snort -v
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "ens33".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.17 GRE (Build 199)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.7.4
           Using PCRE version: 8.44 2020-02-12
           Using ZLIB version: 1.2.8

Commencing packet processing (pid=70155)
```

CONFIGURING SNORT TO RUN IN NIDS MODE

Since we do not want Snort to run as root, we need to create an unprivileged account and group for the daemon to run under (snort: snort). We will also create several files and directories required by Snort, and set permissions on those files. Snort will have the following directories: Configurations and rule files in /etc/snort, Alerts will be written to /var/log/snort, Compiled rules (.so rules) will be stored in /usr/local/lib/snort_dynamicrules

 # CREATE THE SNORT USER AND GROUP:
sudo groupadd snort
sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort
 # CREATE THE SNORT DIRECTORIES:
sudo mkdir /etc/snort
sudo mkdir /etc/snort/rules
sudo mkdir /etc/snort/rules/iplists
sudo mkdir /etc/snort/preproc_rules
sudo mkdir /usr/local/lib/snort_dynamicrules

sudo mkdir /etc/snort/so_rules

 # CREATE SOME FILES THAT STORES RULES AND IP LISTS

sudo touch /etc/snort/rules/iplists/black_list.rules
sudo touch /etc/snort/rules/iplists/white_list.rules
sudo touch /etc/snort/rules/local.rules
sudo touch /etc/snort/sid-msg.map

 # CREATE OUR LOGGING DIRECTORIES:

sudo mkdir /var/log/snort
sudo mkdir /var/log/snort/archived_logs

 # ADJUST PERMISSIONS:

sudo chmod -R 5775 /etc/snort
sudo chmod -R 5775 /var/log/snort
sudo chmod -R 5775 /var/log/snort/archived_logs
sudo chmod -R 5775 /etc/snort/so_rules
sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules

We want to change ownership of the files that have been created above as well to make sure Snort can access the files it uses:

 # CHANGE OWNERSHIP ON FOLDERS:

sudo chown -R snort:snort /etc/snort
sudo chown -R snort:snort /var/log/snort
sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules

Snort needs some configuration files and the dynamic preprocessors copied from the Snort source tarball into the /etc/snort folder.

The configuration files are:

- classification.config
- file_magic.conf
- reference.config
- snort.conf
- threshold.conf
- attribute table.dtd
- gen-msg.map
- unicode.map

To copy the configuration files and the dynamic preprocessors, run the following commands:

cd ~/snort_src/snort-2.9.17.0/etc/
sudo cp *.conf* /etc/snort
sudo cp *.map /etc/snort
sudo cp *.dtd /etc/snort
cd ~/snort_src/snort-2.9.9.0/src/dynamic-preprocessors/build/usr/local/lib/snort_dynamicpreprocessor/
sudo cp * /usr/local/lib/snort_dynamicpreprocessor/

We now have the following directory layout and file locations:
Snort binary file: /usr/local/bin/snort
Snort configuration file: /etc/snort/snort.conf
Snort log data directory: /var/log/snort
Snort rules directories: /etc/snort/rules
/etc/snort/so_rules
/etc/snort/preproc_rules
/usr/local/lib/snort_dynamicrules
Snort IP list directories: /etc/snort/rules/iplists
Snort dynamic preprocessors: /usr/local/lib/snort_dynamicpreprocessor/

Our Snort directory listing looks like this:

```
root@ubuntu:/home/vignesh/Downloads# tree /etc/snort/
/etc/snort/
├── attribute_table.dtd
├── barnyard2.conf
├── classification.config
├── file_magic.conf
├── gen-msg.map
├── preproc_rules
│   ├── decoder.rules
│   ├── preprocessor.rules
│   └── sensitive-data.rules
├── reference.config
├── rules
│   ├── app-detect.rules
│   ├── attack-responses.rules
│   ├── backdoor.rules
│   ├── bad-traffic.rules
│   ├── blacklist.rules
│   ├── botnet-cnc.rules
│   ├── browser-chrome.rules
│   ├── browser-firefox.rules
│   ├── browser-ie.rules
```

We now need to edit Snort's main configuration file, /etc/snort/snort.conf. When we run Snort with this file as an argument, it tells Snort to run in NIDS mode.
We need to comment out all the individual rule files referenced in the Snort configuration file. Instead of downloading each file individually, we can use PulledPork to manage our rulesets, which combines all the rules into a single file. The following line will comment out all rulesets in our snort.conf file (there are about 100 lines to comment out, beginning at line 540):
sudo sed -i "s/include \$RULE_PATH/#include \$RULE_PATH/" /etc/snort/snort.conf
We will now manually change some settings in the snort.conf file, using your favorite editor:
sudo nano /etc/snort/snort.conf
Change the following lines to meet your environment:
Line 45, HOME_NET should match the internal network. In the below example, the HOME_NET is 192.168.228.0/24 with a 24-bit subnet mask (255.255.255.0):
ipvar HOME_NET 192.168.228.0/24

Note: It is not advised to set EXTERNAL NET to !$HOME NET as recommended in some guides since it can cause Snort to miss alerts.
Note: It is vital that the HOME_NET match the IP subnet of the interface that you want Snort to listen on. The command ifconfig | grep "inet add" is used to ensure that we hold the right address and mask set. Often this will be a 192.168.1.x or 10.0.0.x IP address.

Set the following file paths in snort.conf, beginning at line 104:
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/iplists
var BLACK_LIST_PATH /etc/snort/rules/iplists

To make testing Snort easy, we want to enable the local.rules file, where we can add rules that Snort can alert on. Un-comment (remove the hash symbol) from line 546 so it looks like this:
include $RULE_PATH/local.rules

Once the configuration file is ready, we will have Snort verify that it is a valid file and all necessary files it references are correct. We use the -T flag to test the configuration file, the -c flag to tell Snort which configuration file to use, and -i to specify the interface that Snort will listen on (this is a new requirement beginning with the 2.9.8.x version of Snort when an active response is enabled). Run sudo snort -T -c /etc/snort/snort.conf -i ens33. Run this command as shown below and look for the following output

```
        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.17 GRE (Build 199)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.7.4
           Using PCRE version: 8.44 2020-02-12
           Using ZLIB version: 1.2.8

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_S7COMMPLUS  Version 1.0  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: appid  Version 1.1  <Build 5>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>

Snort successfully validated the configuration!
Snort exiting
```

INSTALLING BARNYARD2

It is resource-intensive for Snort to write events in human-readable mode, either to the console or to text files, as it is previously performed. Ideally, the Snort events will be stored in a MySQL database to be viewed, searched, and profile the events. To efficiently get Snort events into a MySQL database, we use Barnyard2. We configure Snort to output events in binary form to a folder and then have Barnyard2 read those events asynchronously and insert them to our MySQL database.

First, install the Barnyard2 pre-requisites:

sudo apt-get install -y mysql-server libmysqlclient-dev mysql-client autoconf libtool

The install will prompt you to create a root mysql user password. For the examples below, we will use snortpass. You should choose something different and more secure and store it safely. We will also be creating a MySQL user account named snort, and the password for that account will be snortpass; please note the difference between these two MySQL accounts and passwords. We need to tell snort that it should output its alerts in a binary format (to a file) that Barnyard2 can process. To do that, edit the /etc/snort/snort.conf file, and after line 521 (the commented line starting with the hash sign), add the following line:

output unified2: filename snort.u2, limit 128

So that lines 521 and 522 now looks like:

 # output unified2: Filename merged.log, limit 128, nostamp, mpls event types, vlan_event_types}

output unified2: Filename snort.u2, limit 128

This line tells Snort to save the alerts in a File called snort.u2.nnnnnnnnnn, where the files are 128 MB before rolling over to a new file (with nnnnnnnnnn being the Unix Epoch time). Note that we do not include: nostamp, mpls_event_types, vlan_event_types as recommended. The reason is that we do want the time stamp on the file, and unless you are working with vlans or mpls, this information can cause issues with certain barnyard2 output modules (unless specifically configured) and Splunk (if using). Adding those two options causes Snort to output version 2 of the unified alert in the unified2 file, which cannot be parsed by some 3rd party tools at this time.

NOW DOWNLOAD AND INSTALL BARNYARD2 2.1.14:

cd ~/snort_src

wget https://github.com/firnsy/barnyard2/archive/master.tar.gz -O barnyard2-Master.tar.gz

tar zxvf barnyard2-Master.tar.gz

cd barnyard2-master

autoreconf -fvi -I ./m4

Barnyard2 needs access to the dnet.h library, which has been installed with the Ubuntu libdumbnet package earlier. However, Barnyard2 expects a different file name for this library. Create a soft link from dnet.h to dubmnet.h so, there are no issues:

sudo ln -s /usr/include/dumbnet.h /usr/include/dnet.h

sudo ldconfig

Depending on the OS version (x86 or x86_64), you need to point the install to the correct MySQL library. Run one of the following two lines to configure the build process, depending on the architecture (if you are unsure which architecture you are running, use the uname -m command:

CHOOSE ONE OF THESE TWO COMMANDS TO RUN
./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu
./configure --with-mysql --with-mysql-libraries=/usr/lib/i386-linux-gnu

Now complete the build and install Barnyard2 to /usr/local/bin/barnyard2:
make
sudo make install

Test Barnyard2 to make sure it installed correctly:

```
root@ubuntu:/home/vignesh/Downloads# /usr/local/bin/barnyard2 -v

  ______     -*> Barnyard2 <*-
 / ,,_  \    Version 2.1.14 (Build 337)
 |o"  )~|    By Ian Firns (SecurixLive): http://www.securixlive.com/
 + '''' +    (C) Copyright 2008-2013 Ian Firns <firnsy@securixlive.com>

USAGE: /usr/local/bin/barnyard2 [-options] <filter options>
```

Once Barnyard2 is installed, the next step is to copy and create some files that Barnyard2 requires to run:
sudo cp ~/snort_src/barnyard2-master/etc/barnyard2.conf /etc/snort/
 # THE /VAR/LOG/BARNYARD2 FOLDER IS NEVER USED OR REFERENCED
 # BUT BARNYARD2 WILL ERROR WITHOUT IT EXISTING
sudo mkdir /var/log/barnyard2
sudo chown snort.snort /var/log/barnyard2
sudo touch /var/log/snort/barnyard2.waldo
sudo chown snort.snort /var/log/snort/barnyard2.waldo
Since Barnyard2 saves alerts to our MySQL database, we need to create that database, as well as a `snort' MySQL user to access that database. Run the following commands to create the database and MySQL user. When prompted for a password, use the snortpass. You will also be setting the MySQL snort user password in the fourth mysql command (to snortpass), so change it there as well.
$ *mysql -u root -p*
mysql> create database snort;
mysql> use snort;
mysql> source ~/snort_src/barnyard2-master/schemas/create_mysql
mysql> CREATE USER 'snort'@'localhost' IDENTIFIED BY 'snortpass';
mysql> grant create, insert, select, delete, update on snort.* to 'snort'@'localhost';
mysql> exit
We need to tell Barnyard2 how to connect to the MySQL database. Edit /etc/snort/barnyard2.conf, and at the end of the file, add this line (changing the password to the one you created above):
output database: log, mysql, user=snort password=snortpass dbname=snort host=localhost sensor name=sensor01
Since the password is stored in cleartext in the barnyard2.conf file, it will prevent other users from reading it:
sudo chmod o-r /etc/snort/barnyard2.conf

CREATING STARTUP SCRIPTS

We want to create startup scripts for Snort and Barnyard2 that will launch the services on system startup. Ubuntu 16 uses the systemD init system, while previous versions of Ubuntu use the Upstart system.

SYSTEMD STARTUP SCRIPT - UBUNTU 16

Ubuntu 16 has moved to systemD for services / daemons. To create the Snort systemD service, use an editor to create a service file:

sudo vi /lib/systemd/system/snort.service

and enter the following content (change the interface name from ens160 if different on your system):

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

[Service]
Type=simple
ExecStart=/usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i ens160

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

Now we tell systemD that the service should be started at boot:

sudo systemctl enable snort

Finally, we want to start the service:

sudo systemctl start snort

To check that the service is running:

systemctl status snort

Next, create the Barnyard2 systemd service. We will add two flags here: -D to run as a daemon, and -a /var/log/snort/archived logs, this will move logs that Barnyard2 has processed to the /var/log/snort/archived/ folder. Use an editor to create a service file:

sudo vi /lib/systemd/system/barnyard2.service

with the following content (the exec content line should be one line, through ...archived_logs):

```
[Unit]
Description=Barnyard2 Daemon
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -q -w /var/log/
snort/barnyard2.waldo -g snort -u snort -D -a /var/log/snort/archived_logs

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

Now we tell systemD that the service should be started at boot:

sudo systemctl enable barnyard2

Finally, we want to start the service:

sudo systemctl start barnyard2

To check that the service is running:

systemctl status barnyard2

Reboot and verify that both services start correctly.

BASE - A WEB GUI FOR SNORT

BASE is a simple web GUI for Snort. Alternate products include Snorby, Splunk, Sguil, AlienVault OSSIM, and any syslog server.

There is a slight difference between BASE on Ubuntu 14 versus 16: BASE requires PHP 5, which is not available in the Ubuntu 16 archives (Ubuntu has moved on to PHP 7 in this release), so we have to use a PPA on Ubuntu 16 to install the php 5 packages

UBUNTU 16 ONLY:

sudo add-apt-repository ppa:ondrej/php
sudo apt-get update
sudo apt-get install -y apache2 libapache2-mod-php5.6 php5.6-mysql php5.6-cli php5.6 php5.6-common php5.6-gd php5.6-cli php-pear php5.6-xml

NEXT INSTALL PEAR IMAGE GRAPH:

sudo pear install -f --alldeps Image_Graph

DOWNLOAD AND INSTALL ADODB:

cd ~/snort_src
wget https://sourceforge.net/projects/adodb/files/adodb-php5-only/adodb-520-for-php5/adodb-5.20.8.tar.gz
tar -xvzf adodb-5.20.8.tar.gz
sudo mv adodb5 /var/adodb
sudo chmod -R 755 /var/adodb

DOWNLOAD BASE AND COPY TO APACHE ROOT

cd ~/snort_src
wget http://sourceforge.net/projects/secureideas/files/BASE/base-1.4.5/base-1.4.5.tar.gz
tar xzvf base-1.4.5.tar.gz
sudo mv base-1.4.5 /var/www/html/base/

CREATE THE BASE CONFIGURATION FILE:

cd /var/www/html/base
sudo cp base_conf.php.dist base_conf.php

Now edit the config file:

sudo vi /var/www/html/base/base_conf.php

with the following settings (note that the trailing slash on line 80 is required, despite the instructions in the configuration file):
$BASE_urlpath = '/base'; # line 50
$DBlib_path = '/var/adodb/'; #line 80
$alert_dbname = 'snort'; # line 102
$alert_host = 'localhost';
$alert_port = '';
$alert_user = 'snort';
$alert_password = 'snortpass'; # line 106

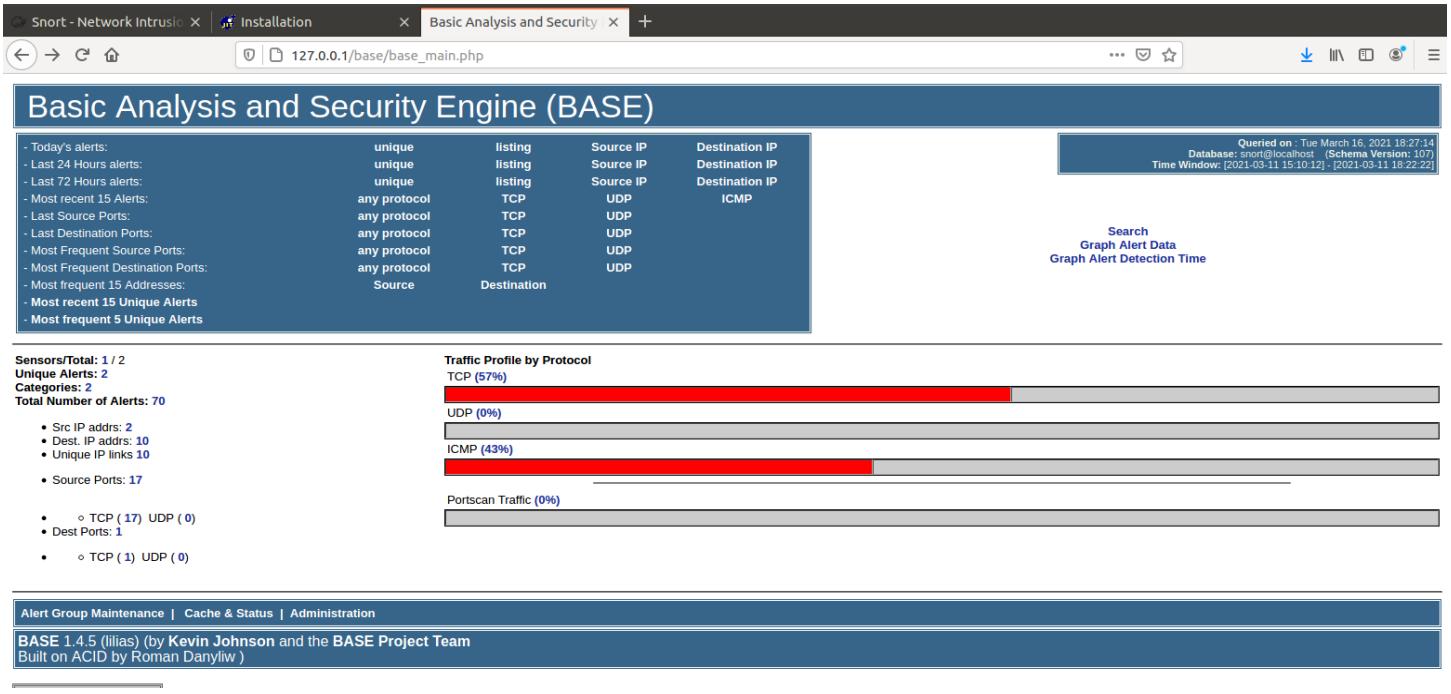
While in the base_conf.php file, you will also want to comment out line 457 (we do not want the DejaVuSansfont), and un-comment (remove the two backslashes) from line 459, enabling a blank font. The section for fonts beginning at line 456) should look like this:
//$graph_font_name = "Verdana";
//$graph_font_name = "DejaVuSans";
//$graph_font_name = "Image_Graph_Font";

$graph_font_name = "";

Set permissions on the BASE folder, and since the password is in the base_conf.php file, we should prevent other users from reading it
sudo chown -R www-data: www-data /var/www/html/base
sudo chmod o-r /var/www/html/base/base_conf.php

RESTART APACHE:
sudo service apache2 restart

The last step to configure BASE is done via http:
1. Browse to http://ServerIP/base/index.php and click on setup page link (replace ServerIP with the IP of your Snort Server).
2. Click on Create BASE AG button on the upper right of the page
3. Click on the Main page line
Note: If you read through the BASE configuration file, there are several other options you can implement if you like, a few to note are: SMTP Email alerts, IP Address to Country Support, and user authentication.
If you have issues, there is a good chance they are related to Barnyard2.



PART III: MANAGE

WRITING A SIMPLE RULE TO TEST SNORT DETECTION

At this stage, Snort does not have any rules loaded (our rule files referenced in snort.conf are empty). You can verify that Snort has not loaded any rules if you scroll up through the output from the previous command and look for: 0 Snort rules read. To test Snort's detection abilities, create a simple rule that will cause Snort to generate an alert whenever Snort sees an ICMP \Echo request" or \Echo reply" message, which is easy to generate with the ubiquitous ping utility (this makes for easy testing of the rule). Paste the following single line into the empty local rules file: /etc/snort/rules/local.rules (note, this should go on one line): alert icmp any any -> $HOME_NET any (msg:"ICMP test detected"; gid:1; sid:10000001; rev:001; classtype:icmp-event;)
Barnyard2 does not read meta-information about alerts from the local.rules file. Without this information, Barnyard2 will not know any details about the rule that triggered the alert, and will generate non-fatal errors

when adding new rules with PulledPork (done in a later step). To make sure that barnyard2 knows that the rule we created with unique identifier 10000001 has the message "ICMP Test Detected" and some other information. We add the following line to the /etc/snort/sid-msg.map file:
gid,1 || sid,10000001 || ref,001 || classification, icmp-event || priority,0 || msg, ICMP Test detected || url, tools.ietf.org/html/rfc792
When you un-commented line 546 above (include $RULE PATH/local.rules), you were telling Snort that the local.rules file should be loaded by Snort. When Snort loads that file on start-up, it will see the rule that has been created and used that rule on all traffic the interface sees. In this case, when we created the rule, we told Snort that it should generate an alert when it sees an ICMP ping.

Since we made changes to the Snort configuration, we should test the configuration file again:
sudo snort -T -c /etc/snort/snort.conf -i ens33
This time if you scroll up through the output, you will find that one rule (the one we created in local.rules and loaded by the include directive in snort.conf) has been loaded:

```
++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
2 Snort rules read
    2 detection rules
    0 decoder rules
    0 preprocessor rules
2 Option Chains linked into 2 Chain Headers
++++++++++++++++++++++++++++++++++++++++++++++++++

+-------------------[Rule Port Counts]---------------------------------------
|             tcp     udp    icmp      ip
|     src       0       0       0       0
|     dst       1       0       0       0
|     any       0       0       1       0
|      nc       1       0       1       0
|     s+d       0       0       0       0
+------------------------------------------------------------------------------

+-----------------------[detection-filter-config]--------------------------------
| memory-cap : 1048576 bytes
+-----------------------[detection-filter-rules]---------------------------------
| none
--------------------------------------------------------------------------------
```

Now that the Snort correctly loads our rule and our configuration, snort can be started in NIDS mode, and tell it to output any alerts right to the console. The Snort will be run from the command line, using the following flags:

-A console	the `console' option prints fast mode alerts to stdout
-q	Quiet mode. Do not show banner and status report.
-u snort	Run Snort as the following user after Startup
-g snort	Run Snort as the following group after Startup
-c /etc/snort/snort.conf	The path to our snort.conf file
-i eth0	The interface to listen on (change to your interface if different)

$ sudo /usr/local/bin/snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33
When you execute this command, you will not initially see any output. Snort is running and is processing all packets that arrive on ens33 (or whichever interface you specified with the -i flag). Snort compares each packet to the rules it has loaded (in this case our single ICMP Ping rule) and will then print an alert to the console when a packet matches our rule.

```
root@ubuntu:/home/vignesh/Downloads# /usr/local/bin/snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33
03/16-17:34:23.573511  [**] [1:10000001:1] ICMP Test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/16-17:34:24.565149  [**] [1:10000001:1] ICMP Test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/16-17:34:25.567513  [**] [1:10000001:1] ICMP Test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/16-17:34:26.643447  [**] [1:10000001:1] ICMP Test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/16-17:34:27.591103  [**] [1:10000001:1] ICMP Test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/16-17:34:28.578628  [**] [1:10000001:1] ICMP Test detected [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
```

From another terminal, ping the IP address of google DNS server 8.8.8.8 on the Snort computer, and the console has output like what is displayed above. This output is the individual alerts that Snort writes to the console when it matches packets to the ICMP rule you created. In the above example, the Snort server is listening on ens33 with an IP address of 192.168.228.139, and the computer generating the ping is 8.8.8.8. Use ctrl-c to stop Snort from running. Note that Snort has saved a copy of this information in /var/log/snort, with the name snort.log.nnnnnnnnn (the numbers may be different). At this point, Snort is running correctly in NIDS mode and generating alerts.

Now that the Snort is tested for writing events to the correct binary log file, and that Barnyard2 is reading those logs and writing the events to our MySQL database. It could just start both programs up in daemon mode and generate some events by pinging the interface (triggering the rule we created earlier), but it is better to test one portion at a time. Run Snort in alert mode (the command we run below is how Snort will usually be run when we set it up as a daemon, except we are not using the -D flag which causes it to run as a daemon).

sudo /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i ens33

Ping the Google Public DNS server, you will not see any output on the screen because Snort was not started with the -A console flag like before. Once the ping stops, type ctrl-c to stop Snort. you should see a new file in the /var/log/snort directory with following name: snort.u2.nnnnnnnnnn (the numbers will be different because they are based on the current time. The snort.log.nnnnnnnnnn is the output file we created when we first tested Snort. You can delete that file if you want:

```
root@ubuntu:/home/vignesh/Downloads# ls -l /var/log/snort/
total 20
drwsrwxr-t 2 snort snort 4096 Mar 11 13:05 archived_logs
-rw-r--r-- 1 root  root     0 Mar 11 15:08 barnyard2.waldo
-rw------- 1 snort snort 1620 Mar 16 17:34 snort.log.1615941242
-rw------- 1 snort snort 1164 Mar 11 15:10 snort.u2.1615504157
-rw------- 1 snort snort  450 Mar 11 16:20 snort.u2.1615508391
-rw------- 1 snort snort 2100 Mar 11 18:22 snort.u2.1615514273
```

We now run Barnyard2 and tell it to process the events in snort.u2.nnnnnnnnnn and load them into the Snort database. We use the following flags with Barnyard2:

-c /etc/snort/barnyard2.conf	The path to the barnyard2.conf file
-d /var/log/snort	The folder to look for Snort output files
-f snort.u2	The Filename to look for in the above directory (snort.u2.nnnnnnnnnn)
-w /var/log/snort/barnyard2.waldo	The location of the waldo file (bookmark file)
-u snort	Run Barnyard2 as the following user after Startup
-g snort	Run Barnyard2 as the following group after Startup

RUN BARNYARD2 WITH THE FOLLOWING COMMAND:

/usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -v -d /var/log/snort/ -f snort.u2

Barnyard2 will start up (be patient, it can take some time), and then it will process the alerts in the /var/log/snort/snort.u2.nnnnnnnnnn file, write them to both the screen and the database, and then wait for more events to appear in the /var/log/snort directory. use Ctrl-c to stop the process. When Barnyard2 is processing the events, you should see output like:

```
      --== Initialization Complete ==--

 ______   -*> Barnyard2 <*-
/ ,,_  \  Version 2.1.14 (Build 337)
|o"  )~|  By Ian Firns (SecurixLive): http://www.securixlive.com/
+ ''''' +  (C) Copyright 2008-2013 Ian Firns <firnsy@securixlive.com>

Opened spool file '/var/log/snort//snort.u2.1615504157'
03/11-16:10:12.498187  [**] [1:10000001:1] Snort Alert [1:10000001:1] [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/11-16:10:13.501705  [**] [1:10000001:1] Snort Alert [1:10000001:1] [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/11-16:10:14.504099  [**] [1:10000001:1] Snort Alert [1:10000001:1] [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/11-16:10:15.505301  [**] [1:10000001:1] Snort Alert [1:10000001:1] [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/11-16:10:16.518019  [**] [1:10000001:1] Snort Alert [1:10000001:1] [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
03/11-16:10:17.509890  [**] [1:10000001:1] Snort Alert [1:10000001:1] [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.228.139
Closing spool file '/var/log/snort//snort.u2.1615504157'. Read 12 records
Opened spool file '/var/log/snort//snort.u2.1615508391'
03/11-17:20:12.822086  [**] [1:10000002:2] Snort Alert [1:10000002:2] [**] [Classification: A TCP Connection was Detected] [Priority: 4] {TCP} 192.168.228.139:60326 -> 163
.172.177.144:80
03/11-17:20:12.822150  [**] [1:10000002:2] Snort Alert [1:10000002:2] [**] [Classification: A TCP Connection was Detected] [Priority: 4] {TCP} 192.168.228.139:60328 -> 163
.172.177.144:80
03/11-17:20:12.822194  [**] [1:10000002:2] Snort Alert [1:10000002:2] [**] [Classification: A TCP Connection was Detected] [Priority: 4] {TCP} 192.168.228.139:60322 -> 163
.172.177.144:80
Closing spool file '/var/log/snort//snort.u2.1615508391'. Read 6 records
Opened spool file '/var/log/snort//snort.u2.1615514273'
03/11-19:06:36.364957  [**] [1:10000002:2] Snort Alert [1:10000002:2] [**] [Classification: A TCP Connection was Detected] [Priority: 4] {TCP} 192.168.228.139:36012 -> 91.
```

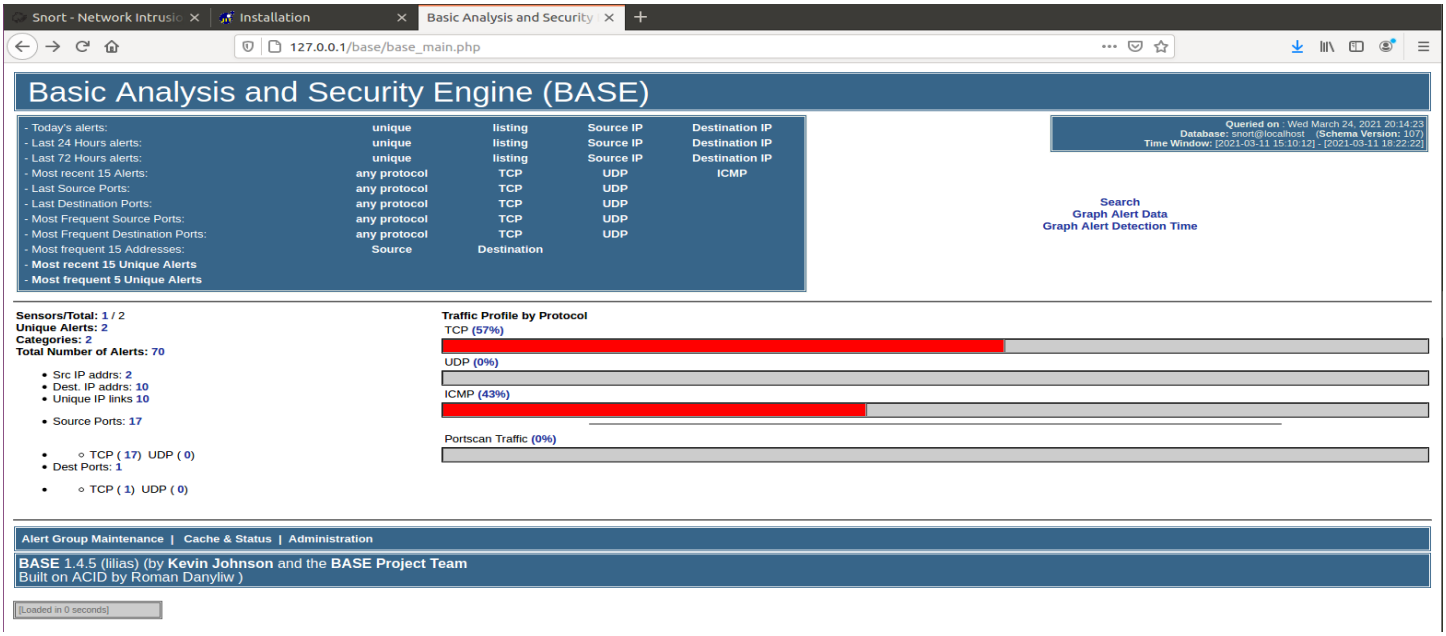
Once the Ctrl-c is pressed to stop barnyard2, it will print information about the records it processed. We now want to check the MySQL database to see if Barnyard2 wrote the events. Run the following command to query the MySQL database, you will be prompted for the MySQL snort user password: snortpass (not the MySQL root password):
mysql -u snort -p -D snort -e "select count(*) from event"

If successful, you will then get the following output, showing the 70 events written to the database from the ICMP request and reply packets.

```
root@ubuntu:/# mysql -u snort -p -D snort -e "select count(*) from event"
Enter password:
+----------+
| count(*) |
+----------+
|       70 |
+----------+
root@ubuntu:/#
```

Congratulations, if you have similar output (count greater than 0) as above, then Snort and Barnyard2 are properly installed and configured. We will create startup scripts later to launch both applications as daemons automatically on boot up.

Then After the Mysql Server has successfully loaded all the rules, now the BASE engine can be loaded to view the alerts in the web browser. In this webpage, both the TCP and ICMP traffic alert were shown, and the appropriate traffic details were also given.



If we view the TCP alert traffic, then it shows much more details as well that includes Source and Destination IP addresses involved in the connection and other details as well.