Snort On Debian

Apr 14, 2014 / Karim Elatov / [iptables](https://elatov.github.io/pages/tags/#iptables-ref), [mysql](https://elatov.github.io/pages/tags/#mysql-ref), [linux](https://elatov.github.io/pages/tags/#linux-ref), [debian](https://elatov.github.io/pages/tags/#debian-ref), [barnyard2](https://elatov.github.io/pages/tags/#barnyard2-ref), [dd\_wrt](https://elatov.github.io/pages/tags/#dd_wrt-ref), [snorby](https://elatov.github.io/pages/tags/#snorby-ref), [snort](https://elatov.github.io/pages/tags/#snort-ref)

* [Snort Version on Debian](https://elatov.github.io/2014/04/snort-debian/#snort-version-on-debian)
* [Compile Snort](https://elatov.github.io/2014/04/snort-debian/#compile-snort)
* [Get Snort Rules with PulledPork](https://elatov.github.io/2014/04/snort-debian/#get-snort-rules-with-pulledpork)
* [Install Barnyard2 for Snort](https://elatov.github.io/2014/04/snort-debian/#install-barnyard2-for-snort)
* [Install Snorby for Snort](https://elatov.github.io/2014/04/snort-debian/#install-snorby-for-snort)
* [Configure Snort Service](https://elatov.github.io/2014/04/snort-debian/#configure-snort-service)
* [Barnyard2 Init Script](https://elatov.github.io/2014/04/snort-debian/#barnyard2-init-script)
* [Move Snort and Barnyard Logs to Dedicated files](https://elatov.github.io/2014/04/snort-debian/#move-snort-and-barnyard-logs-to-dedicated-files)
* [Initial Snort Alerts](https://elatov.github.io/2014/04/snort-debian/#initial-snort-alerts)
* [Forward Traffic to Snort Sensor](https://elatov.github.io/2014/04/snort-debian/#forward-traffic-to-snort-sensor)
* [First Interesting Alert](https://elatov.github.io/2014/04/snort-debian/#first-interesting-alert)

I decided to install Snort on Debian. By default the debian apt sources do have a snort package but it’s out of date.

Snort Version on Debian

Here is the is version available for Debian:

elatov@kerch:~$apt-cache showpkg snort

Package: snort

Versions:

2.9.2.2-3 (/var/lib/apt/lists/ftp.us.debian.org\_debian\_dists\_wheezy\_main\_binary-amd64\_Packages) (/var/lib/apt/lists/ftp.fr.debian.org\_debian\_dists\_wheezy\_main\_binary-amd64\_Packages)

So it’s at 2.9.2.2 but that’s already end of lifed as per [pulledpork](http://blog.snort.org/2012/08/snort-2922-is-end-of-life.html) to download the rules for that old version. So let’s build it from source.

Compile Snort

First let’s install the prerequisites:

elatov@kerch:~$sudo apt-get install flex bison libpcap-dev libdnet-dev libdumbnet-dev

Now let’s get the source:

elatov@kerch:~$wget http://sourceforge.net/projects/snort/files/snort/daq-2.0.2.tar.gz

elatov@kerch:~$wget http://sourceforge.net/projects/snort/files/snort/snort-2.9.6.0.tar.gz

Let’s build the DAQ software. Extract the source:

elatov@kerch:~$tar xvzf daq-2.0.2.tar.gz

Now let’s prepare the source:

elatov@kerch:~$cd daq-2.0.2

elatov@kerch:~$./configure --prefix=/usr/local/snort

Now let’s compile the software:

elatov@kerch:~$make

If the compile is successful, let’s install it:

elatov@kerch:~$sudo mkdir /usr/local/snort

elatov@kerch:~$sudo chown elatov:elatov /usr/local/snort

elatov@kerch:~$make install

Now let’s do the same thing for snort

elatov@kerch:~$tar xvzf snort-2.9.6.0.tar.gz

elatov@kerch:~$cd snort-2.9.6.0

elatov@kerch:~$./configure --prefix=/usr/local/snort --with-daq-includes=/usr/local/snort/include --with-daq-libraries=/usr/local/snort/lib --enable-sourcefire

elatov@kerch:~$make

elatov@kerch:~$make install

Now let’s copy the initial configuration over:

elatov@kerch:~$mkdir /usr/local/snort/etc

elatov@kerch:~$rsync -avzP snort-2.9.6.0/etc/\*.conf\* /usr/local/snort/etc/.

elatov@kerch:~$rsync -avzP snort-2.9.6.0/etc/\*.map /usr/local/snort/etc/.

Edit the main configuration file **/usr/local/snort/etc/snort.conf** and modify the following:

ipvar HOME\_NET 192.168.0.0/16,10.0.0.0/8

ipvar EXTERNAL\_NET !HOME\_NET

var RULE\_PATH ./rules

var WHITE\_LIST\_PATH ./rules

var BLACK\_LIST\_PATH ./rules

output unified2: filename merged.log, limit 128, mpls\_event\_types, vlan\_event\_types

config logdir: /usr/local/snort/var/log

dynamicpreprocessor directory /usr/local/snort/lib/snort\_dynamicpreprocessor/

dynamicengine /usr/local/snort/lib/snort\_dynamicengine/libsf\_engine.so

dynamicdetection directory /usr/local/snort/lib/snort\_dynamicrules

# comment out the specific rules, lines 547 to 661

#include $RULE\_PATH/app-detect.rules

#include $RULE\_PATH/attack-responses.rules

#include $RULE\_PATH/backdoor.rules

Here is an easy command to delete the rules from the configuration file:

elatov@kerch:~$sed -i '/^include $RULE\_PATH/d' /usr/local/snort/etc/snort.conf

Now let’d add the snort user and group:

elatov@kerch:~$sudo groupadd snort

elatov@kerch:~$sudo useradd -g snort snort

Let’s create the rest of the directories that we will be used by **pulledpork**:

elatov@kerch:~$mkdir /usr/local/snort/etc/rules

elatov@kerch:~$mkdir /usr/local/snort/lib/snort\_dynamicrules

elatov@kerch:~$mkdir /usr/local/snort/etc/rules/iplists

elatov@kerch:~$mkdir -p /usr/local/snort/var/log

elatov@kerch:~$touch /usr/local/snort/etc/rules/local.rules

elatov@kerch:~$touch /usr/local/snort/etc/rules/white\_list.rules

elatov@kerch:~$touch /usr/local/snort/etc/rules/black\_list.rules

Get Snort Rules with PulledPork

First let’s install the prerequisites for the **pulledpork** package:

elatov@kerch:~$sudo apt-get install libcrypt-ssleay-perl liblwp-protocol-https-perl

Now let’s get the package:

elatov@kerch:~$svn checkout http://pulledpork.googlecode.com/svn/trunk/ pulledpork-read-only

Now let’s prepare the installation directory:

elatov@kerch:~$sudo mkdir /usr/local/pp

elatov@kerch:~$sudo chown elatov:elatov /usr/local/pp

elatov@kerch:~$mkdir /usr/local/pp/etc

elatov@kerch:~$mkdir /usr/local/pp/bin

Lastly let’s copy the necessary files:

elatov@kerch:~$rsync -avzP pulledpork-read-only/etc/. /usr/local/pp/etc/.

elatov@kerch:~$rsync -avzP pulledpork-read-only/pulledpork.pl /usr/local/pp/bin/.

Now edit the configuration to fit your needs. Here is what I ended up with:

elatov@kerch:~$grep -Ev '^$|^#' /usr/local/pp/etc/pulledpork.conf

rule\_url=https://www.snort.org/reg-rules/|snortrules-snapshot.tar.gz|xxxx

rule\_url=https://s3.amazonaws.com/snort-org/www/rules/community/|community-rules.tar.gz|Community

rule\_url=http://labs.snort.org/feeds/ip-filter.blf|IPBLACKLIST|xxxxx

ignore=deleted.rules,experimental.rules,local.rules

temp\_path=/tmp

rule\_path=/usr/local/snort/etc/rules/snort.rules

local\_rules=/usr/local/snort/etc/rules/local.rules

sid\_msg=/usr/local/snort/etc/sid-msg.map

sid\_msg\_version=1

sid\_changelog=/usr/local/snort/var/log/sid\_changes.log

sorule\_path=/usr/local/snort/lib/snort\_dynamicrules

snort\_path=/usr/local/snort/bin/snort

config\_path=/usr/local/snort/etc/snort.conf

distro=Debian-6-0

black\_list=/usr/local/snort/etc/rules/iplists/default.blacklist

IPRVersion=/usr/local/snort/etc/rules/iplists

version=0.7.0

The xxxx at the end some of the lines corresponds to your oinkcode. Register at [snort](https://www.snort.org/users/sign_in) to get a oinkcode. There are other rules available as well, for example there is [Emerging Threats](http://doc.emergingthreats.net/). The configuration to download rules from *Emerging Threats* is already in the default **pulledpork** configuration file (you just have to enable them, if you want to use them). I initially left them out, just to get used to the current rules first.

Now run the following to get the rules:

elatov@kerch:~$/usr/local/pp/bin/pulledpork.pl -c /usr/local/pp/etc/pulledpork.conf -l

http://code.google.com/p/pulledpork/

\_\_\_\_\_ \_\_\_\_

`----,\ )

`--==\\ / PulledPork v0.7.0 - Swine Flu!

`--==\\/

.-~~~~-.Y|\\_ Copyright (C) 2009-2013 JJ Cummings

@\_/ / 66\_ cummingsj@gmail.com

| \ \ \_(")

\ /-| ||'--' Rules give me wings!

\_\ \_\\

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Checking latest MD5 for snortrules-snapshot-2960.tar.gz....

Rules tarball download of snortrules-snapshot-2960.tar.gz....

They Match

Done!

Checking latest MD5 for community-rules.tar.gz....

They Match

Done!

IP Blacklist download of http://labs.snort.org/feeds/ip-filter.blf....

Reading IP List...

Prepping rules from snortrules-snapshot-2960.tar.gz for work....

Done!

Prepping rules from community-rules.tar.gz for work....

Done!

Reading rules...

Writing Blacklist File /usr/local/snort/etc/rules/iplists/default.blacklist....

Writing Blacklist Version 946091316 to /usr/local/snort/etc/rules/iplistsIPRVersion.dat....

Use of uninitialized value $bin in -f at /usr/local/pp/bin/pulledpork.pl line 986.

Setting Flowbit State....

Enabled 32 flowbits

Done

Writing /usr/local/snort/etc/rules/snort.rules....

Done

Generating sid-msg.map....

Done

Writing v1 /usr/local/snort/etc/sid-msg.map....

Done

Writing /usr/local/snort/var/log/sid\_changes.log....

Done

Rule Stats...

New:-------20635

Deleted:---0

Enabled Rules:----4854

Dropped Rules:----0

Disabled Rules:---15780

Total Rules:------20634

IP Blacklist Stats...

Total IPs:-----2474

Done

Please review /usr/local/snort/var/log/sid\_changes.log for additional details

Fly Piggy Fly!

Now let’s add the community rules to be parsed by snort:

elatov@kerch:~$echo "include \$RULE\_PATH/snort.rules" >> /usr/local/snort/etc/snort.conf

Now make sure snort can start up up without issues:

elatov@kerch:~$sudo /usr/local/snort/bin/snort -c /usr/local/snort/etc/snort.conf -T

At the end should see this:

Running in Test mode

--== Initializing Snort ==--

Initializing Output Plugins!

Initializing Preprocessors!

Initializing Plug-ins!

Parsing Rules file "/usr/local/snort/etc/snort.conf"

Snort successfully validated the configuration!

Snort exiting

If you want to see what snort is catching, you can run the following as a test:

elatov@kerch:~$ sudo /usr/local/snort/bin/snort -A console -q -u snort -g snort -c /usr/local/snort/etc/snort.conf -i eth0

and you should see alerts that snorts catches:

4/01-16:17:24.811261 [\*\*] [129:12:1] Consecutive TCP small segments exceeding threshold [\*\*] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 216.98.195.98:50932 -> 67.172.135.80:4172

Hit **Ctlr-C** to stop the above process, after you confirm it’s seeing traffic. To automatically grab new rules, we can add the **pulledpork** command to the snort user to be run weekly.

Install Barnyard2 for Snort

To help snort process all the packets it recommended to use Barnyard. Barnyard is a processing software which processes a unified2 format file and stores the results in a MySQL database. This way, snort just logs to a file and doesn’t have to waste cycles writing to a database. Let’s install the software, first let’s get the prerequisites:

elatov@kerch:~$ sudo apt-get install libpcap-dev libmysqld-dev

Now let’s get the source:

elatov@kerch:~$wget http://www.securixlive.com/download/barnyard2/barnyard2-1.9.tar.gz

Now let’s install the software:

elatov@kerch:~$tar xvzf barnyard2-1.9.tar.gz

elatov@kerch:~$cd barnyard2-1.9

elatov@kerch:~$./configure --with-mysql --prefix=/usr/local/by --with-mysql-libraries=/usr/lib/x86\_64-linux-gnu

elatov@kerch:~$make

elatov@kerch:~$sudo mkdir /usr/local/by

elatov@kerch:~$sudo chown elatov:elatov /usr/local/by/

elatov@kerch:~$make install

Now we can configure barnyard2 to store alerts in a MySQL database. Here is how my configuration looked like:

elatov@kerch:~$grep -Ev '^$|^#' /usr/local/by/etc/barnyard2.conf

config reference\_file: /usr/local/snort/etc/reference.config

config classification\_file: /usr/local/snort/etc/classification.config

config gen\_file: /usr/local/snort/etc/gen-msg.map

config sid\_file: /usr/local/snort/etc/sid-msg.map

config logdir: /usr/local/snort/var/log

config hostname: kerch

config interface: eth0

config daemon

config set\_gid: 112

config set\_uid: 1002

config waldo\_file: /usr/local/snort/var/log/barnyard2.waldo

input unified2

output alert\_fast: stdout

output database: log, mysql, user=snorby password=snorby dbname=snorby host=localhost

Keep note of the password you specify and make sure when you create the MySQL database with those specifics.

Create other files that will be used upon start up:

elatov@kerch:~$touch /usr/local/snort/var/log/barnyard2.waldo

Install Snorby for Snort

Snorby is nice and organized UI that allows you to check the alerts that were caught by snort. It runs on Ruby on Rails, so let’s set that up. As always, grab the prerequistes:

elatov@kerch:~$ apt-get install libyaml-dev git-core default-jre imagemagick libmagickwand-dev wkhtmltopdf build-essential libssl-dev libreadline-gplv2-dev zlib1g-dev

linux-headers-amd64 libsqlite3-dev libxslt1-dev libxml2-dev libmysqlclient-dev

libmysql++-dev apache2-prefork-dev libcurl4-openssl-dev ruby ruby-dev

Now let’s install **bundler** and **rails**:

elatov@kerch:~$sudo gem install bundler rails

Now let’s install a specific version of **rake**:

elatov@kerch:~$sudo gem install rake --version=0.9.2

Fetching: rake-0.9.2.gem (100%)

Successfully installed rake-0.9.2

1 gem installed

Installing ri documentation for rake-0.9.2...

Installing RDoc documentation for rake-0.9.2...

Now let’s get the source for snorby:

elatov@kerch:~$git clone http://github.com/Snorby/snorby.git

Cloning into 'snorby'...

remote: Reusing existing pack: 10471, done.

remote: Total 10471 (delta 0), reused 0 (delta 0)

Receiving objects: 100% (10471/10471), 9.91 MiB | 413 KiB/s, done.

Resolving deltas: 100% (4764/4764), done.

Let’s configure the MySQL connection settings:

elatov@kerch:~$cp snorby/config/database.yml.example snorby/config/database.yml

Now edit the **snorby/config/database.yml** file and modify the following:

snorby: &snorby

adapter: mysql

username: snorby

password: "snorby"

host: localhost

Now let’s configure the production configuration of snorby:

elatov@kerch:~$cp snorby/config/snorby\_config.yml.example snorby/config/snorby\_config.yml

Then modify the **snorby/config/snorby\_config.yml** file to have the following:

production:

domain: 'demo.snorby.org'

wkhtmltopdf: /usr/bin/wkhtmltopdf

ssl: false

mailer\_sender: 'snorby@snorby.org'

geoip\_uri: "http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz"

rules:

- ""

authentication\_mode: database

Now let’s install the dependencies necessary for snorby:

elatov@kerch:~$ cd snorby

elatov@kerch:~/snorby$ bundle install

Now let’s create a MySQL database for snorby:

elatov@kerch:~$mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 53935

Server version: 5.5.35-0+wheezy1 (Debian)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database snorby;

Query OK, 1 row affected (0.01 sec)

mysql> grant ALL on snorby.\* to snorby@localhost identified by 'snorby';

Query OK, 0 rows affected (0.00 sec)

mysql> exit

Now let’s setup snorby and let snorby create the necessary MySQL schema:

elatov@kerch:~/snorby$bundle exec rake snorby:setup

No time\_zone specified in snorby\_config.yml; detected time\_zone: US/Mountain

5ff841ca217da8aabcebad4b2762f6b6f6d4a531219ea694873f9589f2ad39574c1ab9ecf7738b9922d34479addc3d5958b37ce04f20422359bef099630d8307

ERROR 1007 (HY000) at line 1: Can't create database 'snorby'; database exists

[datamapper] Finished auto\_upgrade! for :default repository 'snorby'

[~] Adding `index\_timestamp\_cid\_sid` index to the event table

[~] Adding `index\_caches\_ran\_at` index to the caches table

[~] Adding `id` to the event table

[~] Building `aggregated\_events` database view

[~] Building `events\_with\_join` database view

\* Removing old jobs

\* Starting the Snorby worker process.

\* Adding jobs to the queue

Lastly go ahead and start up snorby:

elatov@kerch:~/snorby$bundle exec rails server -e production -b 127.0.0.1

No time\_zone specified in snorby\_config.yml; detected time\_zone: US/Mountain

=> Booting WEBrick

=> Rails 3.1.12 application starting in production on http://127.0.0.1:3000

=> Call with -d to detach

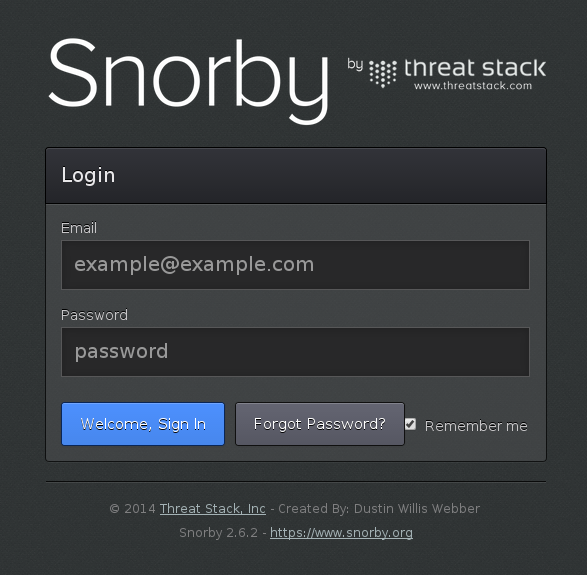
=> Ctrl-C to shutdown server

[2014-03-31 16:04:30] INFO WEBrick 1.3.1

[2014-03-31 16:04:30] INFO ruby 1.9.3 (2012-04-20) [x86\_64-linux]

[2014-03-31 16:04:30] INFO WEBrick::HTTPServer#start: pid=28708 port=3000

At this point you can go to **http://localhost:3000** and see the following:



You can login with:

snorby@snorby.org snorby

You won’t see any alerts in there yet. Lastly let’s make everything owned by the snort user and group:

elatov@kerch:~$sudo chown -R snort:snort /usr/local/snort

elatov@kerch:~$sudo chown -R snort:snort /usr/local/pp

elatov@kerch:~$sudo chown -R snort:snort /usr/local/by

If you want [here](http://groups.google.com/forum/#!topic/snorby/eEDy-FrwHB0) is a link to a start-up script for snorby, but it looks like the recommended approach is to proxy snorby with apache using the passenger plugin. Check out the setup for that [here](http://www.aldeid.com/wiki/Snorby#Recommended_install_of_Passenger).

Configure Snort Service

The snort source contains an init script but it’s for RPM based distros (but we can still make it work). First copy the script:

elatov@kerch:~$sudo cp snort-2.9.6.0/rpm/snortd /etc/init.d/.

Then modify the script to fit your paths (in my case everything was under **/usr/local/snort**) and also add the following on top:

### BEGIN INIT INFO

# Provides: snortd

# Required-Start: $remote\_fs $syslog

# Required-Stop: $remote\_fs $syslog

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# X-Interactive: true

# Short-Description: Start Snort

### END INIT INFO

After that you will be able to add it as service:

elatov@kerch:~$sudo update-rc.d snortd defaults

update-rc.d: using dependency based boot sequencing

You can also use the script from the snort package which is in the aptitude sources. Whatever you do, copy the default configuration for the init script:

elatov@kerch:~$sudo cp snort-2.9.6.0/rpm/snort.sysconfig /etc/default/snort

And enter your configurations there:

ALERTMODE=

CONF=/usr/local/snort/etc/snort.conf

LOGDIR=/usr/local/snort/var/log

Where I start the service here is how it looked like:

elatov@kerch:~$sudo service snortd start

[info] Starting Network Intrusion Detection System snort.

[ ok ] using /usr/local/snort/etc/snort.conf ...done).

Here are the parameters that were passed to the snort daemon:

elatov@kerch:~$ps -eaf | grep snort

snort 2652 1 7 18:29 ? 00:00:00 /usr/local/snort/bin/snort -u snort -g snort -c /usr/local/snort/etc/snort.conf -D -i eth0

I didn’t pass the Log Directory to the daemon, since it was in the **/usr/local/snort/etc/snort.conf** file and I was using unified2 format.

Barnyard2 Init Script

The source for that also had init scripts but they were for RPM. So let’s copy the necessary files:

elatov@kerch:~$sudo cp barnyard2-1.9/rpm/barnyard2 /etc/init.d/.

elatov@kerch:~$sudo cp barnyard2-1.9/rpm/barnyard2.config /etc/default/barnyard2

Do the same thing in the **/etc/init.d/barnyard2** file and add the following to it:

### BEGIN INIT INFO

# Provides: barnyard2

# Required-Start: $remote\_fs $syslog

# Required-Stop: $remote\_fs $syslog

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: Start Barnyard

and fix the paths to fit your local install. Here were the entries I had in my **/etc/default/barnyard2** file:

LOG\_FILE="merged.log"

SNORTDIR="/usr/local/snort/var/log"

INTERFACES="eth0"

CONF=/usr/local/by/etc/barnyard2.conf

EXTRA\_ARGS=""

I was able to add the service without issues:

elatov@kerch:~$sudo update-rc.d barnyard2 defaults

update-rc.d: using dependency based boot sequencing

then here is how the service start looked like:

elatov@kerch:~$sudo service barnyard2 start

Starting Snort Output Processor (barnyard2): Running in Continuous mode

--== Initializing Barnyard2 ==--

Initializing Input Plugins!

Initializing Output Plugins!

Parsing config file "/usr/local/by/etc/barnyard2.conf"

After it started I saw the following process running:

elatov@kerch:~$ps -eaf | grep barnya

snort 2843 1 0 18:40 ? 00:00:00 /usr/local/by/bin/barnyard2 -c /usr/local/by/etc/barnyard2.conf -a /usr/local/snort/var/log/archive -f merged.log -d /usr/local/snort/var/log

Move Snort and Barnyard Logs to Dedicated files

The daemon related logs from both of those services went into **/var/log/daemon.log** and I wanted to separate them out for ease of finding errors. So I added the following into my **rsyslog** config:

elatov@kerch:~$cat /etc/rsyslog.d/by.conf

if $programname == 'barnyard2' then /var/log/barnyard.log

& ~

elatov@kerch:~$cat /etc/rsyslog.d/snort.conf

if $programname == 'snort' then /var/log/snort.log

& ~

To apply the above, restart the rsyslog service:

elatov@kerch:~$ sudo service rsyslog restart

After that if I restarted **barnyard2** , I could just check out the following log to make sure it’s working properly:

elatov@kerch:~$tail -4 /var/log/barnyard.log

Apr 6 18:40:09 kerch barnyard2: Barnyard2 initialization completed successfully (pid=2843)

Apr 6 18:40:09 kerch barnyard2: Using waldo file '/usr/local/snort/var/log/barnyard2.waldo':#012 spool directory = /usr/local/snort/var/log#012 spool filebase = merged.log#012 time\_stamp = 1396830547#012 record\_idx = 25

Apr 6 18:40:09 kerch barnyard2: Opened spool file '/usr/local/snort/var/log/merged.log.1396830547'

Apr 6 18:40:09 kerch barnyard2: Waiting for new data

And same thing for the snort start up:

elatov@kerch:~$tail /var/log/snort.log

Apr 6 18:29:07 kerch snort[2652]: PID path stat checked out ok, PID path set to /var/run/

Apr 6 18:29:07 kerch snort[2652]: Writing PID "2652" to file "/var/run//snort\_eth0.pid"

Apr 6 18:29:07 kerch snort[2652]: Set gid to 112

Apr 6 18:29:07 kerch snort[2652]: Set uid to 1002

Apr 6 18:29:07 kerch snort[2652]:

Apr 6 18:29:07 kerch snort[2652]: --== Initialization Complete ==--

Apr 6 18:29:07 kerch snort[2652]: Commencing packet processing (pid=2652)

You could also check the snort statistic by sending a **USR1** signal to it:

elatov@kerch:~$sudo kill -USR1 19354

Then we should see the following under **/var/log/snort.log**:

Packet I/O Totals:

Received: 12556909

Analyzed: 12556903 (100.000%)

Dropped: 0 ( 0.000%)

Filtered: 0 ( 0.000%)

Outstanding: 6 ( 0.000%)

Injected: 0

Good thing to check to make sure the snort sensor is not overloaded (checking the **Dropped** percentage. Lastly here are my **logrotate** configuration files for each log file:

elatov@kerch:~$cat /etc/logrotate.d/snort

/var/log/snort.log {

daily

rotate 7

compress

missingok

notifempty

create 0640 snort snort

sharedscripts

postrotate

if [ -x /usr/sbin/invoke-rc.d ]; then \

invoke-rc.d snortd restart > /dev/null; \

else \

/etc/init.d/snortd restart > /dev/null; \

fi;

endscript

}

eltov@kerch:~$cat /etc/logrotate.d/barnyard2

/var/log/barnyard2.log {

daily

rotate 7

compress

missingok

notifempty

create 0640 snort snort

sharedscripts

postrotate

if [ -x /usr/sbin/invoke-rc.d ]; then \

invoke-rc.d barnyard2 restart > /dev/null; \

else \

/etc/init.d/barnyard2 restart > /dev/null; \

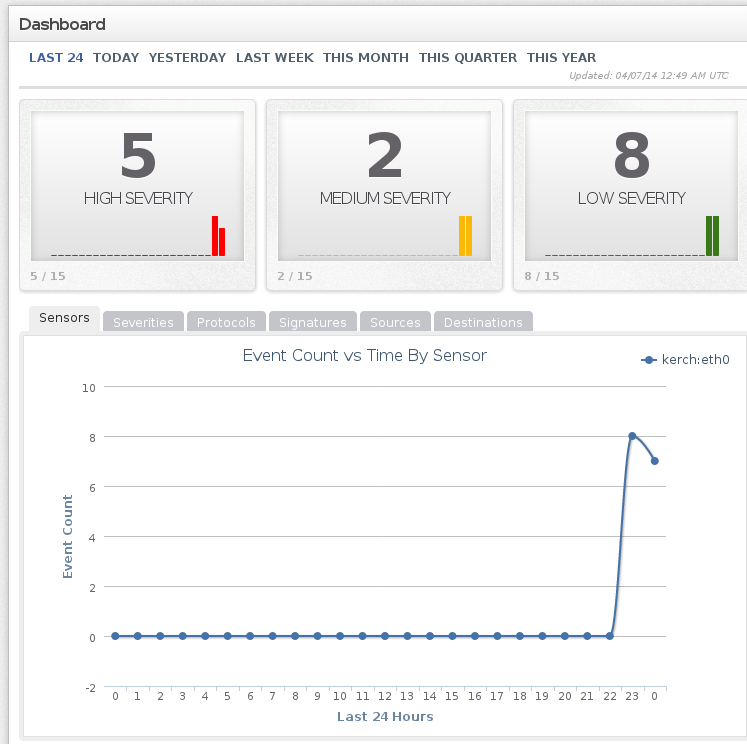
fi;

endscript

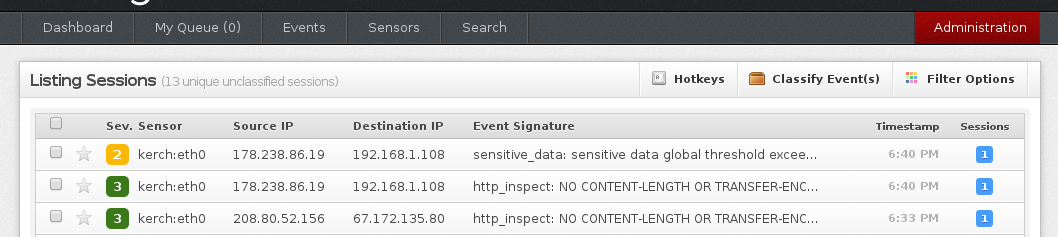
}

Initial Snort Alerts

After some time if you login to snorby, you should see some alerts:



If you go to the events tab, you will see the specifics:



I was getting a bunch of false positive initially, here are some rules I added to suppress some of them:

# Suppress the "stream5 tcp small segment threshold"

suppress gen\_id 129, sig\_id 12, track by\_src, ip 192.168.1.0/24

# Suppress the "stream5 reset outside window"

suppress gen\_id 129 ,sig\_id 15

# Suppress the "ssh: Protocol mismatch"

suppress gen\_id 128, sig\_id 4, track by\_dst, ip 192.168.1.0/24

# Suppress the http\_inspect: UNKNOWN METHOD"

suppress gen\_id 119 ,sig\_id 31

These all went into the **/usr/local/snort/etc/rules/local.rules** file. I also disabled the DNP3 pre-processer (I was getting the following messages **dnp3: DNP3 Link-Layer Frame was dropped**), since I wasn’t part of such a network. This is done by commenting out the following under the **/usr/local/snort/etc/snort.conf** file:

#preprocessor dnp3: ports { 20000 } \

# memcap 262144 \

# check\_crc

Both of the above required a **service snortd restart**.

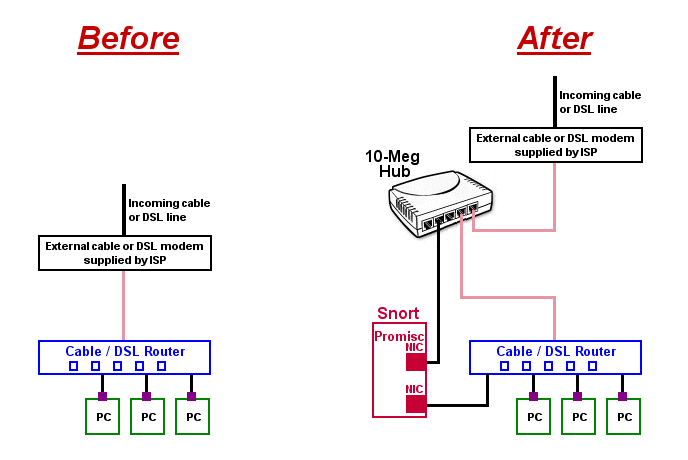
Forward Traffic to Snort Sensor

The best thing to do, would be to put a switch between your Cable Modem and your Router (if you are at home) and then the snort machine would see all the packets. I didn’t want to do that, so I setup my dd-wrt router to forward all the packets to a specific IP. This can be accomplished by logging into the dd-wrt router and running the following:

iptables -A PREROUTING -t mangle -j ROUTE --gw 192.168.1.x --tee

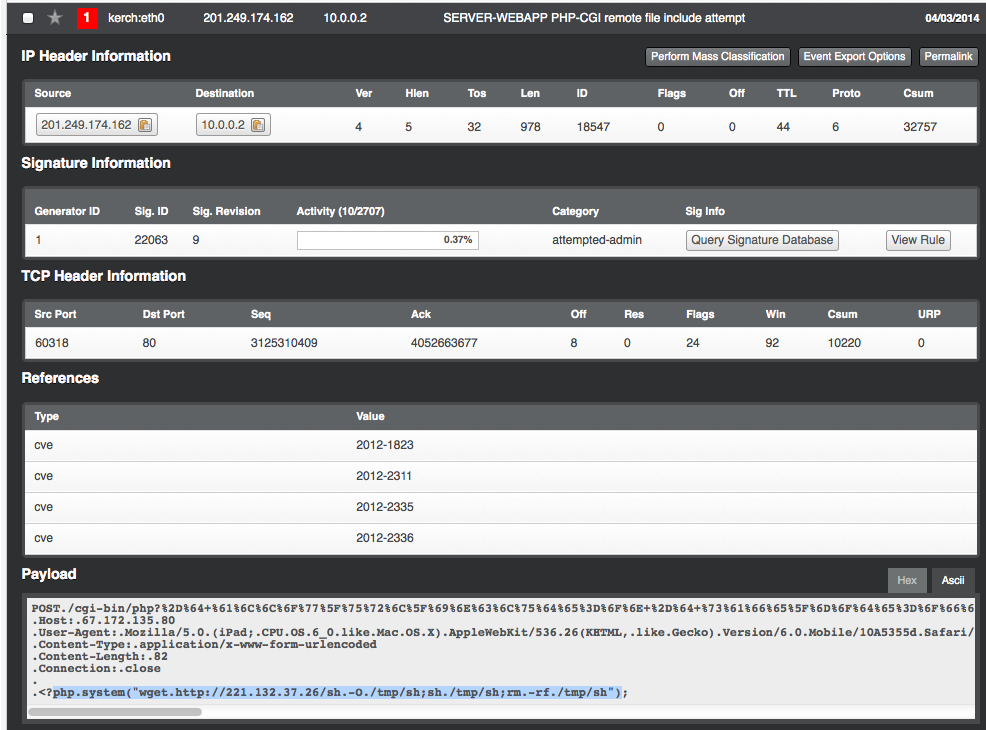
iptables -A POSTROUTING -t mangle -j ROUTE --gw 192.168.1.x --tee

This is not recommended for performance reasons. I kept an eye on my DD-WRT router and I didn’t see any performance issues. If the router starts to bog down, I will try to setup the other recommended configuration. BTW from [this](http://www.aboutdebian.com/snort.htm) site, here is suggested approach:



First Interesting Alert

After a couple of days, I saw the following alert:



[Here](http://thembits.blogspot.com/2014/02/php-cgi-exploitation-never-dies.html) is a little more information about the attack and here is a [link](https://www.netsparker.com/blog/web-security/remote-file-inclusion-vulnerability/) that talks about disabling PHP Remote File Inclusion.