Snorting the Enterprise

(Enterprise Snort Installations)

Introduction

****This document is not complete. Snort 1.8 is still in beta and several issues with the overall design of the system need work. Be aware that parts of the system are broken. I am releasing this document now to get some feedback and for people that might be working on something similar.****

Snort is a great Network Intrusion Detection System (NIDS). Unfortunately, there aren't instructions on how to set things up when your environment is large and distributed. I have made an attempt to document my use of Snort and this is the result of my experiences in an enterprise environment. This document can easily be used to duplicate my environment or setup something similar on a smaller scale. The intent is to have a master console with numerous remote sensors. The sensors are contacted by the console for securely gathering Snort data and pushing out new rules. I have standardized on certain applications and the examples below reflect those choices. You can probably use these instructions on any platform you choose. I welcome feedback and hope you find it helpful.

Overview

The goal of this design is to have multiple remote sensors that log to a central console server. The console is where data crunching takes place. This is achieved through sensors logging in tcpdump format, once an hour those logs are picked up by the console. The console then replays the tcpdump files and inserts that info into ACID. ACID running on the console allows the logs to be viewed with a web browser. The sensors are streamlined so they are only capturing and logging data. All the processing of logs and analysis of data takes place on the console server. My goal was making the sensors fast, so viewing logs on each sensor is not demonstrated here. This document has not been written for the first time Linux or Snort user. It may be useful to someone just starting out, but a minimum level of understanding is assumed.

Environment

It is important to note my design is the result of my environment. Each sensor monitors one subnet. If you are monitoring multiple subnets you may run into problems. I haven't done any testing on sensors that log multiple subnets. It will probably work, but I haven't tested it. Make sure you understand what is happening and realize this document can't be for everyone. It will be helpful for the majority of people and you may get some valuable info from it.

Requirements

I use several tools to achieve the goal of automating as much as possible. Kudos to the authors of the various applications, they did the work, I just put it all together.

Redhat 7.1 - http://www.redhat.com
Snort - http://www.snort.org
Snort Daily from CVS http://snort.sourcerforge.net
Libpcap 0.6.2 - http://www.tcpdump.org
OpenSSH - http://www.openssh.com
Snorticus - http://snorticus.baysoft.net (I severely destroyed Paul's scripts to make them do what I wanted. They aren't pretty.)
ACID - http://www.cert.org/kb/acid/
ADODB - http://php.weblogs.com/adodb
PHP - http://www.php.net
MySQL - http://www.mysql.com
Apache - http://www.apache.org

Getting Started
The sensors and the console need to have Redhat 7.1 and Snort 1.8-beta7 (Build 27) installed. Installing the OS is beyond the scope of the document, so it is assumed you have installed Redhat 7.1. While I have the commands listed below to install each application, it is important to read the INSTALL and README files for each one. Snort 1.8-beta7 (Build 27) may not be stable at this point, I haven't had any problem. Build 27 is the minimum build, all newer builds should work. Older versions of snort **may** work, but they haven't been tested. This document will be updated when 1.8 is officially released.

Sensor Install

All the sensors will be the same, so you can duplicate these instructions for each one. I haven't scripted the install process, but it is possible. As a side note, all my snort boxes are the using the same hardware and will use identical builds. The easy way to create a large number of identical servers is with imaging. I used System Imager to build my boxes quickly. Check it out here: http://systemimager.org. SystemImager also allows me to push updates to the boxes via OpenSSH. This is not related to snort rules; it is specific to OS installs, and system updates. It is an easy way to securely keep sensors up to date.

Securing the Sensor
The only way to connect to the sensor should be through OpenSSH. Remove applications that aren't being used. This not only secures the box, but also saves drive space for logging. SCP can be used in place of FTP, which is commonly left open for file transfers.

OpenSSH
RedHat 7.1 has SSH installed by default. Make sure telnet is disabled.

The first step is to configure OpenSSH to allow the console to pull data from sensor without a password.

Edit the SSHD config file.
vi /etc/ssh/sshd_config

Change the following settings:
Protocol 1,2
IgnoreRhosts no
IgnoreUserKnownHosts yes
RhostsAuthentication no
RhostsRSAAuthentication yes

Edit /root/.shosts and add the console IP and the user that will be connecting.
vi .shosts
console root

Edit /etc/hosts and hard code the console name and IP
Console 192.168.120.5

/etc/rc.d/init.d/sshd restart

We need to get the console's key onto the sensor. If you have the key you can cut and paste it.
ssh console

Answer yes to the next question.

The authenticity of host ' sensor ' can't be established.
RSA key fingerprint is

b2:fc:62:5e:5f:06:ca:63:0a:8c:52:03:db:ad:be:52.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added ' sensor,192.168.138.21' (RSA)
to the list of known hosts.

Now we add the the key to ssh_known_hosts.
cd /root/.ssh
cat known_hosts >> /etc/ssh/ssh_known_hosts

For each sensor, you will need to ssh from the console to the sensor and save the key. You will also need to add the sensor's key to the console's known_hosts file. There is step-by-step instructions in the console section on how to do this.

Libpcap
Snort uses libpcap and it needs to be installed prior to installing snort. In order to support multiple interfaces, the latest version of libpcap needs to be used. Get the latest version here. http://www.tcpdump.org I used libpcap-0.6.2 and the examples below will use that version.

#Let's install libpcap
tar -zxvf libpcap-0.6.2.tar.gz
cd libpcap-0.6.2
./configure
make
make install

Snort
Snort 1.8-beta7 (Build 27) is the best snort yet. Plug-in support and the rule system make it feature rich. The installation instructions below are for a default install of snort.

tar -xvzf snort-1.8.tar.gz
cd snort-1.8
./configure
make
make install

We need to create a user for running snort.
useradd -d /var/log/snort -c "snort user account" -s /bin/false snort

The start script for the sensors is configured to log in tcpdump format. The console will retrieve the tcpdump file and replay it locally and insert the data into ACID. If you need to troubleshoot make sure you are using the correct command line. The easiest way is to remove the –D from the snort command line in hourly.sh. This will prevent snort from starting in daemon mode and allow you to see the errors on the screen.

You can add additional logging to the sensor rules file for archiving or for running things like SnortSnarf on each sensor. My goal was to have sensors that only logged and logged quickly, so it isn't included here. It is as simple as modifying the rules file with extra logging.

Script
I started out with Paul Ritchey's Snorticus scripts. He was using them to move data from the sensor to the console for SnortSnarf. Paul did all the hard work figuring out the site system and dealing with multiple networks. I modified them to move the tcpdump files from the sensors to the console and then have console replay the tcpdump files through snort.

We need to create the following directories.
mkdir /var/log/snort

mkdir /var/log/snort/LOGS
mkdir /var/log/snort/LOGS/MYSITE1
mkdir /var/log/snort/rules
mkdir /var/log/snort/scripts

Make sure snort owns everything.
chown -Rf snort:snort /var/log/snort

Create the script below.
vi /var/log/snort/scripts/hourly.sh
See Appendix 1 for the script.

You will also need to modify several variables in hourly.sh.
- sensor_site: This is the used by the sensor and the console to group sensors by site. Example: MySite1 Example 2: MySite2
- data_expires: This is the number of days to keep old data. Example: 30
- rules_directory: This is the rules directory we created. /var/log/snort/rules
- log_directory: This is where we log snort data. /var/log/snort/LOGS
- network_list_file: The network configuration file has the info about what subnets this sensor monitors. Use the full path to the file. /var/log/snort/rules/network.cfg

We want this script to be run hourly, so it needs to be in root's crontab.

crontab -e

Add the following line.
*01 * * * * /var/log/snort/scripts/hourly.sh > /dev/null 2>&1*

We need to create the network.cfg file. If your network is 192.168.1.0 then your file will look like the following. There is a pipe symbol between the IP and the interface name. Every network you are monitoring is on its own line.

vi /var/log/snort/rules/network.cfg
192.168.1.0|eth0

Now we need to get some rules. The directory where we untar'ed snort has some sample rules.

cp snort.conf /var/log/snort/rules/rules.192.168.120.0
*cp *.rules /var/log/snort/rules/*
cp classification.config /var/log/snort/rules/

You can manually kick off snort by using the command line in the crontab.

/var/log/snort/scripts/hourly.sh

After you have started it, check to see if snort is running by typing.
ps –ef|grep snort

You should see snort running. If not, time to go troubleshoot. The easiest way is to remove the –D from the snort command line in hourly.sh. With the –D removed, snort will not start in Daemon mode. Error messages will be written to the screen and make troubleshooting easier.

Console

For the most part, the console is identical to the sensors. In addition to the sensor build there are a few other apps installed. I included each step here to make it clear what needs to be installed on the console. Several apps have dependencies, so it is important to install in the order shown.

OpenSSH

Here is where we configure the console for connecting to the sensor without a password.

vi /etc/ssh/ssh_config
Change the Protocol line to
Protocol 1,2

Connect to the sensor to save the key.
ssh sensor

Answer yes to the next question.

The authenticity of host ' sensor ' can't be established.
RSA key fingerprint is
b2:fc:62:5e:5f:06:ca:63:0a:8c:52:03:db:ad:be:52.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added ' sensor,192.168.138.21' (RSA)
to the list of known hosts.

Now we add the the key to ssh_known_hosts.
cd /root/.ssh
cat known_hosts >> /etc/ssh/ssh_known_hosts

MySQL

This will need to be installed prior to Snort.

tar -xvzf mysql-3.23.39.tar.gz
cd mysql-3.23.39
./configure
make
make install
scripts/mysql_install_db
safe_mysqld &
mysqladmin -u root password abc123
cp support-files/mysql.server /etc/rc.d/init.d/mysqld
chmod 755 /etc/rc.d/init.d/mysqld
ln -s ../init.d/mysqld /etc/rc.d/rc3.d/S85mysqld
ln -s ../init.d/mysqld /etc/rc.d/rc3.d/K85mysqld
/etc/rc.d/rc3.d/S85mysqld start

Snort install

The difference between the sensor install and the console is the addition of the MySQL support.

tar -xvzf snort-1.8.tar.gz
cd snort-1.8
./configure --with-mysql=/usr/local
make
make install

The following command will fix any MySQL errors you might have when you try to start snort.
ldconfig /usr/lib /usr/X11R6/lib /usr/local/lib /usr/local/lib/mysql

Apache
Here is the Apache install config I use.

tar -zxvf apache_1.3.20.tar.gz
cd apache_1.3.20
./configure \
--prefix=/usr/local/apache \
--enable-module=most \
--enable-shared=max
make
make install

Open the Apache config file
vi /usr/local/apache/conf/httpd.conf
Uncomment the line with ServerName and put in either your servers IP or its DNS name.

cp /usr/local/apache/bin/apachectl /etc/rc.d/init.d/
ln -s ../init.d/apachectl /etc/rc.d/rc3.d/S87httpd
ln -s ../init.d/apachectl /etc/rc.d/rc3.d/K87httpd
/etc/rc.d/rc3.d/S87httpd start

PHP
To install PHP

tar -zxvf php-4.0.5.tar.gz
cd php-4.0.5
./configure \
--with-apxs=/usr/local/apache/bin/apxs \
--with-config-file-path=/usr/local/apache/conf \
--enable-versioning \
--enable-bcmath \
--with-mysql \
--enable-ftp \
--disable-debug \
--enable-memory-limit=yes \
--enable-track-vars
make
make install

ACID
The console runs ACID. The latest version acid-0.9.6b10.tar.gz is installed on my console. ACID also needs PHP and I use Apache for my webserver, make sure both of these are installed before installing ACID. ACID is not secure don't assume it is. Don't put the console box anywhere that can be accessible from the Internet.

ACID needs ADODB here are instructions for installing it.

cd /usr/local/apache/
tar -zxvf adodb112.tgz
cd adodb

We need to create the database that the master console will log to and ACID will pull info from. Get back to the directory with the snort source.

cd snort

echo "CREATE DATABASE snort;" | mysql -u root -p
mysql -u root -p snort < ./contrib/create_mysql
mysqladmin -u snort password snort
mysql -u root -p < ./contrib/create_mysql

cp acid-0.9.5.tar.gz /home/httpd/html
tar -xvfz acid-0.9.5.tar.gz
cd acid
vi /usr/local/apache/htdocs/acid/acid_conf.php

Modify the following variables for your setup
$DBlib_path = "/usr/local/apache/adodb";
$DBtype = "mysql";

$alert_dbname = "snort";
$alert_host = "localhost";
$alert_port = "";
$alert_user = "snort";
$alert_password = "snort";

$archive_dbname = "snort";
$archive_host = "localhost";
$archive_port = "";
$archive_user = "snort";
$archive_password = "snort";

mysql -u root -p snort </usr/local/apache/htdocs/acid/create_acid_tbls_mysql.sql
mysql -u root –p
grant INSERT,SELECT on snort. to snort@localhost IDENTIFIED BY 'snort';*

http://your.web.server/acid/. If things are working correctly, you will see the ACID front page. If not, you should refer to the appropriate software package for help.

Scripts
Modified Snorticus scripts automate the process of moving tcpdump files and rule changes from sensors to the console. See the sensor script section for more info.

mkdir /var/log/snort
mkdir /var/log/snort/LOGS
mkdir /var/log/snort/rules
mkdir /var/log/snort/scripts

Now let's create the script for restarting Snort and compressing the logs from the previous hour.
vi /var/log/snort/scripts/retrieve.sh

See script 2 in the Appendix

You will also need to modify several variables in retrieve.sh for your install.
- sensor_site: This is the used by the sensor and the console to group sensors by site. Example: Site1 Example 2: Site2
- log_directory: Directory for the logs. /var/log/snort/LOGS
- data_expires: This is the number of days to keep old data. Example: 30
- rules_directory: This is the rules directory we created. /var/log/snort/rules
- log_directory: This is where we log snort data. /var/log/snort/
- network_list_file: The network configuration file has the info about what subnets this sensor monitors. Use the full path to the file. /var/log/snort/rules/network.cfg

We want this script to be run fifteen minutes after the hour, so it needs to be in root's crontab. Add the following line for each sensor.

*15 * * * * /var/log/snort/scripts/retrieve.sh MYSITE1 sensor1.packetnexus.com > /dev/null 2>&1*

It is important that the crontab has exactly what is in each hourly.sh script. MYSITE1 is different than MySite1.

We need to create the network.cfg file. If are monitoring 192.168.120.0 and 10.10.0.0 then your file will look like the following. There is a pipe symbol between the IP and the interface name. Every network you are monitoring is on its own line.

vi /var/log/snort/rules/network.cfg
192.168.120.0|eth0
10.10.0.0|eth0

Now we need to get some rules. The snort directory has some sample rules.

cp snort.conf /var/log/snort/rules/rules.192.168.120.0
*cp *.rules /var/log/snort/rules/*

Testing
Testing the entries in the crontab is difficult. You can manually kick off the sensor by using the command in root's crontab. Testing the sensors is not as easy. One thing to do if things aren't working is to remove the –D switch and test the scripts. If there is an error you will see it on your screen.

Deployment
Installing the sensors should be easy. They should be configured before they are installed on the network they are monitoring. If things are correct, things will work as planned. Knock on wood.

Summary
There are a couple of added bonuses with this design. It is fast, so might be able to get by with less hardware. Logging of packets in tcpdump format allows you to recreate an attack. That could come in handy if you catch something. The process of the console pulling the data from the sensors allows you to use strict ACL's for access to the sensors. You can configure the sensors to only accept SSH connections from the console. This will probably make it into future versions.

There are some things this setup does not do. The console doesn't have up to the minute attack info and it doesn't have any method of alerting you to attacks in progress. Hopefully the next version of this document will include both of the features above. I also plan to add support for updating the rules on each sensor from the master console. At this time, port scan data is not collected. Future versions of this document will include that info.

Jason Lewis
http://www.packetnexus.com/

This document is Copyright (c) 2001, Jason Lewis. All rights reserved. Permission to distribute the document is hereby granted providing that distribution is electronic, no money is involved, reasonable attempts are made to use the latest version and all credits and this copyright notice are maintained. Other requests for distribution will be considered. All reasonable requests will be granted.

Appendix

These scripts are available here:

http://www.packetnexus.com/docs/packetnexus/SnortEntScripts.tar.gz

1.Hourly.sh

```
#!/bin/csh -fx
#
# /var/log/snort/scripts/hourly.sh
#
# Copyright (C) 2000 Paul Ritchey <pritchey@arl.army.mil>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

# Brief Description:
# This script is designed to be run through a cron job
# once an hour, every hour.

# It's purpose is to stop/restart the snort sensor and
# tar/gzip hourly results.

# Version History:
#
# 10/05/2000     Version 1.0.1   Removed pkill, killall and
#                                   made it much more generic.
#                                   Fixed bug in bit determination
#                                   for CIDR notation.
#                                   You can now use multiple NIC
#                                   ports by specifying them in
#                                   the network.cfg file with
#                                   the ip address.
#                                   Paul Ritchey, JASI
#
# 08/03/2000        Version 1.0     First version of script.
#                                   Paul Ritchey, JASI
#

# BEGIN USER CONFIGURABLE SECTION
# BEGIN USER CONFIGURABLE SECTION
# BEGIN USER CONFIGURABLE SECTION

# This variable should be set to the name of
# the site being monitored.  It is used as part
# of the directory structure for the logs, and
# is also important to the 'pullback' script on
# the analysis box.
set sensor_site='MYSITE1'

# This should be set to the number of days the data
# should reside on this server until it is deleted
# in order to conserve space.
set data_expires=3
```

```
# This variable should be set to the directory
# that contains the rule sets you want used for each
# network block you want snort to watch.
set rules_directory='/var/log/snort/rules'

# This variable should be set to the directory where
# snort should log everything to.  It is the 'root' directory
# where snort will create hourly log directories, and then
# tar/gzip the hourly directories when the hour ends.
set log_directory='/var/log/snort/LOGS'

# This variable should be set to the directory and filename
# of the configuration file containing the network blocks
# snort should watch.
set network_list_file='/var/log/snort/rules/network.cfg'

# END USER CONFIGURABLE SECTION
# END USER CONFIGURABLE SECTION
# END USER CONFIGURABLE SECTION

# Algorithm used:
#      1.  pkill the snort process(es)
#      2.  restart the snort process(es)
#             a.  Retrieve netblocks to watch.
#             b.  Create new directories
#                    1.  Create new hour directory.
#                    2.  Create new netblock subdirectories.:wq
#             c.  Start snort instance for each netblock.
#                    1.  Determine CIDR info for current netblock
#                    2.  Start instance of snort with appropriate params.
#      3.  Tar and gzip the last hour's data so it can be pulled
#          down by the analyst box.
#      4.  Now delete data over a certain age (user specified)


set path=( /usr/bin /usr/sbin /usr/local/bin /bin )

# Due to some differences in the OS commands, we must
# determine the proper settings/commands for Linux.
# They vary slightly between Linux and Solaris.
if (`uname -s` == 'Linux') then
       set tar_flags='cvfP'
       set ps_flags='aux'
else
       # Until others give me the proper settings
       # (if there are any other differences)
       # we assume OS is Solaris (SunOS).
       set tar_flags='cvf'
       set ps_flags='-ef'
endif

# Now go read in the config file containing all of the
# network blocks snort is to watch.
set network_to_watch=(`grep -v '#' $network_list_file`)

# Get the date information to that gets used for the creation of
# the hourly data.
set current_date=`date '+%Y%m%d'`
set current_hour=`date '+%H'`
set current_datehour=$current_date.$current_hour

#  Set up the previous hour date information.
if ($current_hour > '0') then
       set previous_date=$current_date
       @ previous_hour=$current_hour - 1
else
       @ previous_date=$current_date - 1
       set previous_hour=23
endif

# Make sure we have enough digits in the hour.
# Otherwise pad it with a '0'.
```

```
if ($previous_hour < '10') then
       set previous_hour = 0$previous_hour
endif
echo $previous_hour

set previous_datehour=$previous_date.$previous_hour

# Make the hourly logging directory.
mkdir $log_directory/$sensor_site/$current_datehour

#  First we need to kill all running snort processes.
set snort_pids=(`ps "$ps_flags" | grep snort | grep -v grep | grep -v hourly |
tr -s ' ' | sed 's/^[ ^t]*//' | cut -f 2 -d ' ' -`)
set myloop=1
while ( $myloop <= $#snort_pids )
       kill $snort_pids[$myloop]
       @ myloop++
end

# Since all scans are logged to the same log file, move it
# into the proper date directory.
mv $log_directory/$sensor_site/snort_portscan.log
$log_directory/$sensor_site/$previous_datehour

# Now loop through each of the netblocks, creating the appropriate
# directories.
set myloop=1
while ( $myloop <= $#network_to_watch )
       # Break the network information from the config file apart.
       set mynetwork_block = `echo $network_to_watch[$myloop] | cut -f 1 -d '|'
-`
       set mynetwork_port = `echo $network_to_watch[$myloop] | cut -f 2 -d '|' -
`
echo $mynetwork_block
echo $mynetwork_port

       # Make a netblock subdirectory in the hourly logging
       # directory.
       mkdir $log_directory/$sensor_site/$current_datehour/$mynetwork_block

       # Determine CIDR bit count for current netblock.
       set myblock = $mynetwork_block
       set myloop2 = 1
       set mybitcount = 4
       while ( $myloop2 <= 4)
              if ( $myblock:e == '0') then
                     @ mybitcount -=1
              endif

              set myblock = $myblock:r
              @ myloop2++
       end
       @ mybitcount *=8

       # Start an instance of snort for the current netblock.
snort -A full -b -c $rules_directory/rules.$mynetwork_block -d -D -e -h
$mynetwork_block/$mybitcount -i $mynetwork_port -l
$log_directory/$sensor_site/$current_datehour/$mynetwork_block/


       # Now increment our loop counter
       @ myloop++
end

# Now it's time to tar and gzip the previous hours data.
tar $tar_flags $log_directory/$sensor_site/$previous_datehour.tar
$log_directory/$sensor_site/$previous_datehour
gzip $log_directory/$sensor_site/$previous_datehour.tar
rm -r $log_directory/$sensor_site/$previous_datehour

# Now delete data older than the user specified retension period.
find $log_directory/$sensor_site  -mtime $data_expires -exec rm -f {} \;
```

```
# We're done, so exit gracefully.
exit 0
```

2.retrieve.sh

```
#!/bin/csh -f
# This script is designed to be run through a cron job
# once an hour, every hour.

# It's purpose is to retrieve the hourly wrapup from
# the snort sensor to the analysis box.  Called
# without any parameter it will retrieve the
# previous hour's wrapup from the sensor.  Called with
# a parameter it will retrieve the requested hour's
# wrapup data from the sensor.

# After retrieving the hourly wrapup, it will process
# the data (after untarring and ungzipping it) through
# SnortSnarf to create the html pages for the analyst(s).

# Version History:
#
# 10/19/2000        Version 1.0.2   Per report from Steve M., I missed
#                                   the 'P' flag for the tar command
#                                   which is needed under linux.  I
#                                   had it in the hourly_wrapup.sh, but
#                                   missed it here.
#
# 09/29/2000     Version 1.0.1   Per request from Jim Hoagland,
#                                   added ability to specify more
#                                   SnortSnarf options.
#
# 08/04/2000        Version 1.0     First version of script.
#                                   Paul Ritchey, JASI

# BEGIN USER CONFIGURABLE SECTION
# BEGIN USER CONFIGURABLE SECTION
# BEGIN USER CONFIGURABLE SECTION

# This should be set to the directory where
# all sensor data retrieved from the sensor
# is to reside.  This script will automatically
# place the data into the appropriate site
# subdirectory in this directory.  It is also
# used when retrieving data from the sensor.
set log_directory='/var/log/snort/LOGS'

# This should be set to the number of days the data
# should reside on this server until it is deleted
# in order to conserve space.
set data_expires=7

set network_list_file='/var/log/snort/rules/network.cfg'

# This variable should be set to the path where the
# SnortSnarf perl script can be found ('snortsnarf.pl').
# Or you can use this to point to whatever script you
# choose to use for processing the retrieved log data.
set snortsnarf_path='/var/log/snort/scripts'

# This variable should be set to include any extra features
# of snortsnarf that you want to use.
# NOTE:  DO NOT INCLUDE THE -d OR -l OPTIONS, THESE TWO
# OPTIONS ARE ALREADY IN USE.
set snortsnarf_options=''

# OPTIONS ARE ALREADY USED BY THIS SCRIPT.
# This variable should be set to the path where
# the GNU 'date' command resides.
# NOTE:  GNU 'date' COMES WITH LINUX, BUT NOT OTHER
# FLAVORS OF UNIX. WHEN DEALING WITH MULTIPLE PLATFORMS
# YOU MAY WANT TO 'ln -s /usr/bin/date /usr/local/date'
# ON THE LINUX BOXES.
```

```
set gnudate_path='/bin'

# This is the account to use when scp'ing the wrapup
# files from the sensors.
set retrieve_account='root'

# END USER CONFIGURABLE SECTION
# END USER CONFIGURABLE SECTION
# END USER CONFIGURABLE SECTION

# Algorithm used:
#      1.  Make sure we have enough parameters to run with.
#             a. site name (used for directory name on remote box)
#             b. sensor name or ip.
#             c. + | - for time offset
#             d. number to be added/subtracted from local time
#                    NOTE:  ITEMS C AND D WILL BE
#                    IGNORED WHEN USED MANUALLY TO RETRIEVE DATA FROM COMMAND
LINE.
#             e. OPTIONAL: date.hour to retrieve by hand.
#                    Format: yyyymmdd.hh
#                    Example: 20000801.13 retrieves 08/01/2000 at 1300
#      2.  If we passed step 1 ok, go retrieve data.
#             a.  If using 'manual' mode, retrieve the requested hour's
#                 data the user is manually retrieving.
#             b.  If using 'auto' mode, get date/time and retrieve
#                 previous hour's data.
#      3.  Gunzip/untar retrieved data, then delete file.
#      4.  Call script to process data.
#      5.  Delete directories/files over certain age (user specified).


set path=( /usr/bin /usr/sbin /usr/local/bin /bin )

# Due to some differences in the OS commands, we must
# determine the proper settings/commands for Linux.
# They vary slightly between Linux and Solaris.
if (`uname -s` == 'Linux') then
        set tar_flags='xvfP'
else
        # Until others give me the proper settings
        # (if there are any other differences)
        # we assume OS is Solaris (SunOS).
        set tar_flags='xvf'
endif

# Check for number of parameters, we need at least 2 to get started.
if ($#argv < 2) then
       echo ""
       echo "Incorrect number of parameters passed.  Usage:"
       echo ""
       echo "retrieve_wrapup.sh <site_name> <hostname | ip> [yyyymmdd.hh]"
       echo ""
       echo ""
       exit 0
endif

# Now go read in the config file containing all of the
# network blocks snort is to watch.
set network_to_watch=(`grep -v '#' $network_list_file`)


# Determine the date/time we need to retrieve if the user
# hasn't passed it in to us.
if ($#argv > 2) then
       set previous_datehour=$argv[3]
       set previous_date=$argv[3]:r
else
       set previous_datehour=`ssh $retrieve_account@{$argv[2]}
"$gnudate_path/date --date='1 hour ago' '+%Y%m%d.%H'"`
       set previous_date=$previous_datehour:r
endif
```

```
# Test to see if the site logging directory exists
# here on the analyst box.
if !( -e $log_directory/$argv[1] ) then
       mkdir $log_directory/$argv[1]

       # Make sure it was created, if not, notify user
       # and exit.
       if !( -e $log_directory/$argv[1] ) then
              echo " "
              echo "Unable to continue....."
              echo "Unable to create needed site parent directory:"
              echo "    "$log_directory/$argv[1]
              echo " "
              exit 0
       endif
endif

# Now test for the date subdirectory we are retrieving
# data for.
if !( -e $log_directory/$argv[1]/$previous_date ) then
        mkdir $log_directory/$argv[1]/$previous_date

        # Make sure it was created, if not, notify user
        # and exit.
        if !( -e $log_directory/$argv[1]/$previous_date ) then
                echo " "
                echo "Unable to continue....."
                echo "Unable to create needed site date directory:"
                echo "    "$log_directory/$argv[1]/$previous_date
                echo " "
                exit 0
        endif
endif

# Go to the proper directory before we retrieve the sensor data.
cd $log_directory/$argv[1]/$previous_date

# Now retrieve the data from the sensor via scp.
scp
$retrieve_account@{$argv[2]}:$log_directory/$argv[1]/$previous_datehour.tar.gz
$previous_datehour.tar.gz

# Gunzip and untar the file.....
gunzip  -c $previous_datehour.tar.gz | tar $tar_flags -

# Now move the resulting directory into the proper spot.
mv $log_directory/$argv[1]/$previous_datehour
$log_directory/$argv[1]/$previous_date

# Delete the scp'ed file.
rm $previous_datehour.tar.gz

# Now cd into the directory we are processing....
cd $log_directory/$argv[1]/$previous_date/$previous_datehour

# Get a directory listing.  We need to make sure we process all
# subnets that may be at this site.
set subnet_list=`ls`

# Create the directory that will hold ALL of the subnet
# log data.
mkdir snort_logs


# Walk through each item in the directory listing.
set myloop=1
while ($myloop <= $#subnet_list)
       # See if the item is a directory.
       if (-d $subnet_list[$myloop]) then
              cat $subnet_list[$myloop]/alert >> snort.alert
              cat $subnet_list[$myloop]/TELNET >> TELNET
```

```
              cp -rf $subnet_list[$myloop]/* snort_logs
              rm -rf $subnet_list[$myloop]
       endif

       # increment the counter.
       @ myloop++
end

# Delete the bogus snort.alert file that results from doing
# the copies in the while loop
rm -f snort_logs/snort.alert
rm -f snort_logs/TELNET

# Open up the permissions show people can see the log
# entries down at the packet level.
chmod -R 755 snort_logs


# Now loop through each of the netblocks, creating the appropriate
# directories.
set myloop=1
while ( $myloop <= $#network_to_watch )
       # Break the network information from the config file apart.
       set mynetwork_block = `echo $network_to_watch[$myloop] | cut -f 1 -d '|'
-`
echo $mynetwork_block



# The final and last step, process the date and create
# the web pages!!!
mkdir $log_directory/$argv[1]/$previous_datehour
mkdir $log_directory/$argv[1]/$previous_datehour/$mynetwork_block
cd $log_directory/$argv[1]/$previous_datehour/$mynetwork_block



foreach file (snort*)
    /usr/local/bin/snort -u snort -g snort -d -c /etc/snort/snort.conf -r
$log_directory/$argv[1]/$previous_datehour/$mynetwork_block/$file
end

       # Now increment our loop counter
       @ myloop++
end



# Now delete data older than the user specified retension period.
find $log_directory/$argv[1]  -mtime $data_expires -exec rm -rf {} \;

# We're done, so exit gracefully.
exit 0
```