Snort−Setup for Statistics HOWTO

Sandro Poppi

spoppi at gmx.de

v1.01, Feb 23, 2002

Revision History

Revision 1.01	2002−02−23	Revised by: sp
− added "Setting up Linux for Snort" section − added mysql option −p − added some clarifications in mysql section		
Revision 1.0	2002−01−01	Revised by: sp
− first release version − moved to snort version 1.8.3 − changed RPMS to point to www.snort.org − added link for my snortd initscript − added warning about automatic rule update − added hint to IDSPM − changed for rule files to /etc/snort to reflect snort.org's RPMS − as allways: clarified some parts		
Revision 0.05	2001−11−14	Revised by: sp
− renamed HOWTO to Snort−Setup for Statistics HOWTO − added short statistic script which I was inspired by Greg Sarsons − clarified some parts and corrected some typos		
Revision 0.04	2001−09−29	Revised by: sp
− added section "snort internal statistics" suggested from Greg Sarson − added short statistic script contributed by Greg Sarson but commented it out to get a more general version		
Revision 0.03	2001−09−19	Revised by: sp
− added throttle option to swatch.conf − changed ACID to version 0.9.6b15 − added some comments in ACID section − added MD5 checksum section but commented it out		
Revision 0.02	2001−09−16	Revised by: sp
Some clarifications as suggested from Greg Sarsons, thx ;)		
Revision 0.01	2001−09−04	Revised by: sp
Initial version		

This HOWTO describes how to configure Snort version 1.8.3 to be used in conjunction with the statistical tools ACID (Analysis Console for Intrusion Databases) and SnortSnarf. It also intends to get some internal statistics out of snort, e.g. if there are packets dropped.

Additionally a description of how to automatically update Max Vision's rules, some scripts which may be helpful and a demo swatch configuration is included.

Table of Contents

1. Introduction

This document was written when I created an IDS sensor with Snort and using some statistic tools in order to help others implementing it. If at least one out there can be helped it has been worth the work.

Snort is an excellent Network Intrusion Detection System (NIDS) for various unices. The Snort homepage can be found at http://www.snort.org/. The version described here is 1.8.3 which was the actual version at the time of writing.

The statistic tools I will describe here are ACID, a database analysis tool for Snort which can be found at http://www.cert.org/kb/acid/ and SnortSnarf, a statistic tool for Snort logs downloadable from http://www.silicondefense.com/software/snortsnarf/index.htm.

Additional support packages are needed for ACID. These are a PHP4 capable webserver like *apache* (http://www.apache.org/), PHPlot used for creating graphs in PHP (http://www.phplot.com/) and ADODB used for connecting to databases with PHP (http://php.weblogs.com/ADODB/).

The description also includes which additional software is needed for ACID and how to configure along with some scripts I use including a changed version of the snortd initscript and a short chapter about swatch (http://www.stanford.edu/~atkins/swatch) a log file watcher script written in perl. I created a swatch RPM which can be found at http://www.lug−burghausen.org/projects/Snort−Statistics/swatch−3.0.2−1.noarch.rpm.

One hint for those interested in maintaining more than one snort sensor: You might take a look at IDSPM (IDS Policy Manager) at http://www.activeworx.com/ which is an application to maintain various sensors with different policies along with merging capabilities for new rules and a lot more. The only "nasty" thing is that it runs on W2K/XP and is not (yet?) Open Source.

1.1. Copyright Information

This document is copyrighted (c) 2001, 2002 Sandro Poppi and is distributed under the terms of the Linux Documentation Project (LDP) license, stated below.

Unless otherwise stated, Linux HOWTO documents are copyrighted by their respective authors. Linux HOWTO documents may be reproduced and distributed in whole or in part, in any medium physical or electronic, as long as this copyright notice is retained on all copies. Commercial redistribution is allowed and encouraged; however, the author would like to be notified of any such distributions.

All translations, derivative works, or aggregate works incorporating any Linux HOWTO documents must be covered under this copyright notice. That is, you may not produce a derivative work from a HOWTO and impose additional restrictions on its distribution. Exceptions to these rules may be granted under certain conditions; please contact the Linux HOWTO coordinator at the address given below.

In short, we wish to promote dissemination of this information through as many channels as possible. However, we do wish to retain copyright on the HOWTO documents, and would like to be notified of any plans to redistribute the HOWTOs.

If you have any questions, please contact <`linux−howto at metalab.unc.edu`>

1.2. Disclaimer

No liability for the contents of this documents can be accepted. Use the concepts, examples and other content at your own risk. As this is a new edition of this document, there may be errors and inaccuracies, that may of course be damaging to your system. Proceed with caution, and although this is highly unlikely, the author(s) do not take any responsibility for that.

All copyrights are held by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements.

You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

1.3. New Versions

This is the initial release.

The main site for this HOWTO is http://www.lug−burghausen.org/projects/Snort−Statistics/.

Mirrors may be found at the Linux Documentation Project or Snort homepages.

The newest version of this HOWTO will always be made available on the main website, in a variety of formats:

- HTML.
- compressed postscript (A4).
- SGML source.

1.4. Credits

Credits go to a variaty of people including

- Martin Roesch <`roesch at sourcefire.com`> Author of Snort
- Roman Danyliw <`roman at danyliw.com`> Author of ACID
- James Hoagland <`hoagland at SiliconDefense.com`> Author of SnortSnarf
- Stuart Staniford <`stuart at SiliconDefense.com`> Author of SnortSnarf
- Joe McAlerney <`joey at siliconDefense.com`> Author of SnortSnarf
- John Lim <`jlim at natsoft.com.my`> Author of ADODB
- Afan Ottenheimer <`afan at users.sourceforge.net`> Author of PHPlot
- Andreas Östling <`andreaso at it.su.se`> Author of arachnids_upd
- Max Vision <`vision at whitehats.com`> "Distributor" of vision.rules and maintainer of http://www.whitehats.com/
- Greg Sarsons <`gsarsons at home.com`> for proof reading and suggestions
- All the peaople on the *snort−users* mailinglist, they helped me and of course they will help YOU >;)
- ...

If I missed someone it was not because of not honoring her or his work!

1.5. Feedback

Feedback is most certainly welcome for this document. Without your submissions and input, this document wouldn't exist. Please send your additions, comments and criticisms to the following email address : <`spoppi at gmx.de`>.

1.6. Translations

There are currently no translations available.

2. Structure

This document is supposed to be a step by step guide on how to install and configure snort version 1.8.3, ACID, a web based frontend for statistical realtime snort data with the underlying MySQL database and its support packages PHPlot and ADODB, SnortSnarf, also a statistical tool with a web frontend for analysing the snort logfile, arachnids_upd for always getting the actual rules from Max Vision's http://www.whitehats.com/ site, and a sample swatch configuration I use to check if snort reports errors which I do not get because snort has stopped.

3. Technical Overview

Snort is mainly a so called Network Intrusion Detection System (NIDS), it is Open Source and available for a variaty of unices as well as Microsoft Windows (R).

A NIDS cares for a whole network segment in contrast to a host based IDS which only cares for the host it is running on.

Since NIDS are mostly used in conjunction with firewalls it is vital to not being vulnerable for attacks itself. Therefor all interfaces used with snort bound to should be set up without ip addresses. Since this can not be achieved in every configuration, e.g. if you want to bind snort on an isdn interface ippp0, it should be considered to use a standalone computer for snort and set it up as a firewall and router for the dial−up connection too.

For more information on that topic see the *Firewall−HOWTO* or my *Firewalling+Masquerading+Diald+dynamic IP−HOWTO*.

Snort can be used to care for more than one network segment which we will discuss later.

Snort also can be used as a sniffer to troubleshoot network problems, but that's not a topic in this document.

ACID, the Analysis Console for Intrusion Databases, is part of the AIR−CERT project. It makes use of PHPlot, a library for creating nice graphs in PHP, and ADODB, an abstraction library for combining PHP and various database systems like MySQL and PostgreSQL. The ACID homepage says:

"The Analysis Console for Intrusion Databases (ACID) is a PHP−based analysis engine to search and process a database of incidents generated by security−related software such as IDSes and firewalls."

Max Vision's IDS rules (referred to as *vision.rules* because this is the name of the downloadable file) are used to complete the rules shipped with snort.

arachnids_upd is a small but fine perl script which downloads the actual *vision.rules* using *wget* and optionally deletes single rules given in an ASCII file.

4. Configuration

This chapter describes the various configuration tasks to get snort and the tools up and running.

Since I am using RedHat linux 7.x all the given pathnames and configuration options are eventually RedHat specific while there should be no big problem to transfer it to any other distribution.

4.1. Setting up Linux for Snort

Instead of doing the work twice I only provide a link to a document describing the various tasks of compiling/installing MySQL, Apache, ACID etc. by Jason Lewis: http://www.packetnexus.com/docs/packetnexus/

Please keep in mind that I'm not the author of either the document or the scripts mentioned there. I didn't even test the scripts so please don't ask me about them ;)

4.2. Configuring Snort

You can start installing snort by getting the actual tarball from http://www.snort.org/ and compile it yourself or try to find precompiled binaries for your distribution.

For version 1.8.3 you can find precompiled binaries for rpm based linux distributions, FreeBSD, Solaris and Windows at www.snort.org.

I'm no longer maintaining my own RPMS since work hasn't to be done more than once. But I will offer you my adjusted *snortd.multi* initscript at http://www.lug−burghausen.org/projects/Snort−Statistics/snortd.multi.

My old 1.8.1 RPMS with MySQL support (but without PostgreSQL support!) can still be found at http://www.lug−burghausen.org/projects/Snort−Statistics/snort−1.8.1−4.i386.rpm. To create a postgreSQL enabled version, download the Source RPM, edit the spec file and rebuild the RPM. If you are not familiar with creating RPMs you should have a look on the *RPM−HOWTO* or http://www.rpm.org/ where *Maximum RPM* is located, a downloadable book about RPM along with other good sources about RPM.

4.2.1. /etc/snort/snort.conf

After installing the RPM we have to edit */etc/snort/snort.conf* to reflect our needs. Martin Roesch created the Snort Users Manual which is shipped with the snort tarball and the RPMS as a PDF version. You should have a look on it to see which options you would like to use as not all but only the ones needed for our configuration here will be covered in this document.

Also the example configuration */etc/snort/snort.conf* shipped with the tarball/RPM is a good place to start because of the detailed remarks.

4.2.1.1. Snort Variables

First we define various variables like HOME_NET, EXTERNAL_NET and DNS_SERVERS to reflect our network topology. Make sure you use the right addresses or you get weird, or worse, no alarms.

When using snort in a complex environment, let's say one sensor with multiple interfaces to watch, the definition of HOME_NET and EXTERNAL_NET may be hard or at least results in a very long list, you can set both variables to *any*. You loose some kind of pre−filtering for the sake of not having to put in dozens of network ranges in a large internal network. And you minimize the performance impact of having snort run through a huge list of addresses for each packet.

To get rid of some nasty messages of (false) portscans define the variable DNS_SERVERS to hold all ip addresses of dns−servers along with other nodes like network management stations triggering snort's portscan module. This is an ongoing process.

You also can define your own variables here which you can refer to in your own rules. This is helpful e.g. if using *pass rules* to suite your environment.

Define all other variables to appropriate values or as in the shipped */etc/snort/snort.conf* to $HOME_NET.

```
      var HOME_NET any
      var EXTERNAL_NET any
      # DNS_SERVERS holds the addresses of "noisy" computers like DNS or NWM
      # to be ignored from portscans
      var DNS_SERVERS [1.1.1.1/32,2.2.2.2/32]
      var SMTP_SERVERS $HOME_NET
      ...
```

4.2.1.2. Snort Preprocessors

Next we have to set up the preprocessors to be used. While the more preprocessors you use you get more triggers for alarms but for the cost of performance. So be careful in choosing preprocessors.

You should also have a look on Marty's *Snort Users Manual* because some preprocessors are deprecated. For those you should use the new introduced ones.

The preprocessors *minfrag* and *stream* are depricated in favor of *stream4*, and *defrag* is deprecated by *frag2*.

frag2 is the new IP defragmentation processor introduced in snort v1.8 which should be more memory efficient than *defrag/minfrag*.

From the Snort Users Manual: *The stream4 module provides TCP stream reassembly and stateful analysis capabilities to Snort. Robust stream reassembly capabilities allow Snort to ignore "stateless" attacks such as stick and snot produce.Stream4 also gives large scale users the ability to track more than 256 simultaneous TCP streams. Stream4 should be able to scale to handle 64,000 simultaneous TCP connections.*

The *stream4* module consists of two preprocessors called *stream4* and *stream4_reassemble*, which both have to be used.

There are various options for both preprocessors while we will use only − for *stream4* − *detect_scans* for getting alarms for portscan events and *detect_state_problems* to be informed when stream events like evasive RST packets, data on SYN packets and out of window sequence numbers occur.

With *stream4_reassemble* we use the option *ports all* what makes the reassembly catch all ports instead of only some predefined ones. To be honest, this is some kind of paranoic and impacts the cpu utilization of the snort sensor, but since I didn't get any bad results listening on a Pentium III 800 MHz on three 100 Mbit/s full duplex lines with average to low utilization I think it's the better solution.

Two other preprocessors we will use are *portscan* and *portscan−ignorehosts* which are responsible for portscan detection (*portscan*) and for which hosts portscan detection has to be ignored (*portscan−ignorehosts*).

For *portscan* we define to look for every network using the form *0.0.0.0/0*, set the number of port numbers to be accessed in the also to be defined detection period in seconds. Additionally we have to provide the complete path to the portscan logfile.

With *portscan−ignorehosts* we get rid of some weird alarms from hosts which talk too much and trigger portscan detection like name servers and network management stations (see variable *DNS_SERVERS* above).

Some preprocessors which are not (yet) mentioned in Marty's Users Manual but we will use are *unidecode* which is a replacement of *http_decode* and normalizes http and UNICODE attacks, *rpc_decode* to normalize rpc traffic on a given port, *bo* to check for back orifice traffic and *telnet_decode* to normalize telnet negotiation strings.

Other preprocessors like SPADE are not yet covered here but may be in a future version. Contributions are very welcome >;)

After all that theoretical stuff here is the preprocessor part of */etc/snort/snort.conf*:

```
      preprocessor frag2
      preprocessor stream4: detect_scans detect_state_problems
      preprocessor stream4_reassemble: ports all
      preprocessor unidecode: 80 8080
      preprocessor rpc_decode: 111
      preprocessor bo: −nobrute
      preprocessor telnet_decode
      preprocessor portscan: 0.0.0.0/0 6 3 /var/log/snort/portscan.log
      preprocessor portscan−ignorehosts: $DNS_SERVERS
```

4.2.1.3. Snort Output Modules

The next part is the configuration of the output modules of which we will use the syslog module *alert_syslog* to send alerts to syslog and *database* to additionally log to a MySQL database.

The *alert_syslog* module requires some options for what has to be logged. If like in my case you are using SnortSnarf to analyse the logfile you'll have to add the option *LOG_PID* else SnortSnarf has problems.

As stated before we will use ACID and thus we need to set up snort to log to a database. I chose MySQL for no particular reason (well, I've heard more from MySQL than from postgreSQL but that's all).

The *database* output module requires the following parameters:

log | alert

> Log to the *alert* facility. Also possible would be the *log* facility. If you would like to get portscan alerts into the database you have to use *alert* here.

mysql|postgrsql|odbc|oracle|mssql

> This is the type of database.

user=<username>

> Here you define the username to be used with the database.

password=<password>

> The required password for the given user.

dbname=<databasename>

> The name of the database to be used for logging into.

host=<hostname>

> Here you define the host on which the database is running. Use localhost if the database is running on the snort sensor itself.

sensor_name=<sensor name>

> Here you put in a unique name which is used to differentiate between various sensors if more than one is logging into a single database.

Now let's take a look on the output module part of */etc/snort/snort.conf*:

```
 output alert_syslog: LOG_AUTH LOG_ALERT LOG_PID
 output database: alert, mysql, user=snort password=mypassword dbname=snort host=localhost
```

If you are using more than one physical snort sensor and would log to a database I would recommend using a central database on a separate machine. You then can correlate alert data with a single console getting a better overview when attacks are found.

4.2.1.4. Snort Rule Sets

The rules are the vital part of snort. There are various categories of rules shipped with snort. They can be found in */etc/snort/*, ending with **.rules*. The format in version 1.8+ has changed to reflect the classification types. In addition priority settings of the classtypes can also be defined.

If you're using the original snort tarball I suggest copying all rule files and *classification.config* into it.

The configuration of classification types is done in */etc/snort/classification.config*. Normally you don't have to touch it since it is preconfigured for the shipped snort rules. But if you (again like me) are using Max Vision's *vision.rules* you'll have to add some lines because the classtypes are different. Just copy and paste all *config classification:* lines from *vision.conf* to */etc/snort/classification.config*. And remember to take the *vision.rules* for snort 1.8 (called *vision18.rules* and *vision18.conf* on http://www.whitehats.com/) as the older ones are not prepared for the new format introduced in snort 1.8!

Here's the */etc/snort/classification.config* I used with *vision.rules*:

```
       #
       # config classification:shortname,short description,priority
       #
       #config classification: not-suspicious,Not Suspicious Traffic,0
       config classification: unknown,Unknown Traffic,1
       config classification: bad-unknown,Potentially Bad Traffic, 2
       config classification: attempted-recon,Attempted Information Leak,3
       config classification: successful-recon-limited,Information Leak,4
       config classification: successful-recon-largescale,Large Scale Information Leak,5
       config classification: attempted-dos,Attempted Denial of Service,6
       config classification: successful-dos,Denial of Service,7
       config classification: attempted-user,Attempted User Privilege Gain,8
       config classification: unsuccessful-user,Unsuccessful User Privilege Gain,7
       config classification: successful-user,Successful User Privilege Gain,9
       config classification: attempted-admin,Attempted Administrator Privilege Gain,10
       config classification: successful-admin,Successful Administrator Privilege Gain,11

       # added from vision18.conf
       # classification for use with a management interface
       # low risk
       config classification: not-suspicious,policy traffic that is not suspicious,0
       config classification: suspicious,suspicious miscellaneous traffic,1
       config classification: info-failed,failed information gathering attempt,2
       config classification: relay-failed,failed relay attempt,3
       config classification: data-failed,failed data integrity attempt,4
       config classification: system-failed,failed system integrity attempt,5
       config classification: client-failed,failed client integrity attempt,6
       # med risk
       config classification: denialofservice,denial of service,7
       config classification: info-attempt,information gathering attempt,8
       config classification: relay-attempt,relay attempt,9
       config classification: data-attempt,data integrity attempt,10
       config classification: system-attempt,system integrity attempt,11
       config classification: client-attempt,client integrity attempt,12
       config classification: data-or-info-attempt,data integrity or information gathering attemp
       config classification: system-or-info-attempt,system integrity or information gathering at
       config classification: relay-or-info-attempt,relay of information gathering attempt,15
       # high risk
       config classification: info-success,successful information gathering attempt,16
       config classification: relay-success,successful relay attempt,17
       config classification: data-success,successful data integrity attempt,18
       config classification: system-success,successful system integrity attempt,19
       config classification: client-success,successful client integrity attempt,20
```

The classification and rule files are included in */etc/snort/snort.conf*. Some rule files used here have been copied from the CVS, e.g. *virus.rules* because they were not shipped with the standard distribution.

As stated before the *vision.rules* file will be fetched via the tool *arachnids_upd* which is discussed later.

Arachnids_upd changes the name from *vision18.rules* to *vision.rules* but the rules are of course the ones prepared for snort 1.8+.

Since the variable definitions for INTERNAL and EXTERNAL in *vision.rules* are not the same as with the snort rules I use a script to change these names. Take a look at the *arachnids_upd* section below.

```
       # Include classification & priority settings
       include /etc/snort/classification.config

       include /etc/snort/exploit.rules
       include /etc/snort/scan.rules
       include /etc/snort/finger.rules
       include /etc/snort/ftp.rules
       include /etc/snort/telnet.rules
       include /etc/snort/smtp.rules
       include /etc/snort/rpc.rules
       include /etc/snort/rservices.rules
       include /etc/snort/backdoor.rules
       include /etc/snort/dos.rules
       include /etc/snort/ddos.rules
       include /etc/snort/dns.rules
       include /etc/snort/netbios.rules
       include /etc/snort/web−cgi.rules
       include /etc/snort/web−coldfusion.rules
       include /etc/snort/web−frontpage.rules
       include /etc/snort/web−iis.rules
       include /etc/snort/web−misc.rules
       include /etc/snort/sql.rules
       include /etc/snort/x11.rules
       include /etc/snort/icmp.rules
       include /etc/snort/shellcode.rules
       include /etc/snort/misc.rules
       include /etc/snort/policy.rules
       include /etc/snort/info.rules
       #include /etc/snort/icmp−info.rules
       include /etc/snort/virus.rules
       include /etc/snort/local.rules

       # vision.rules will be catched by arachnids_upd
       include /etc/snort/vision.rules
```

When you are done with setting up */etc/snort/snort.conf* you should start snort by calling */etc/rc.d/init.d/snortd start* and correct any errors you get in the log file */var/log/messages* (ignore any database related messages since the database has not been set up at this time, you also may have to document out the output module database). If everything is ok you can go on with configuring the other parts.

4.2.2. /etc/rc.d/init.d/snortd

In */etc/rc.d/init.d/snortd* you should edit at least the line with the interface to be "snort'ed". Replace the definition of *INTERFACE="eth0"* with the interface you use. This can be another ethernet (*ethx*) but also a *pppx* or *ipppx* interface, e.g. if you are using ISDN your definition should be like

```
     INTERFACE="ippp0"
```

If your snort sensor is only listening on one interface it's sufficient to use the shipped snortd initscript. But if you have more than one interface you may be interested in having a look onto the script I extended for exactly that case. Even when you only have one interface but wish to use swatch the way I do you could copy the swatch parts to the shipped snortd script (see the contrib section of the RPM's documentation).

Next you find the mentioned snortd initscript I extended for snort to listen on more than one interface. One could now say that you can also use *any* as an interface name since the underlying *libpcap* makes this possible, but that's not what I intended to use because I'm not interested in "snorting" the local network where the snort sensor is set up. This should − in a secure environment − be a separate network segment with additional security set up, e.g. a firewall for that segment, so sniffing does not make much sense except if you want to sniff attacks targeted to the snort network itself. Even then, if you use more than one sensor concentrated in that segment you only need to set up one but not all of the sensors for protecting the segment.

I added a new function *daemonMult* derived from RedHat's *daemon* function found in */etc/rc.d/init.d/functions* which is capable of starting a program more than once. I sent RedHat a patch for their *daemon* function to introduce a new option *−−mult* which eventually will be added. If that happens the *daemonMult* function will be obsolete and the call to snort would change from *daemonMult ...* to *daemon −−mult ...*. Let's wait and see.

I also changed the subsystem name from snort to snortd to get rid of error messages when rebooting (the killall script on a redhat box depends on the correct name), just a little typo.

With my script you can now define multiple interfaces to be watched on, just use a space separated list with the *INTERFACE* variable, like in the listing shown below.

Some sanity checks are also included to see if the interface to listen on is already up and if there is an IP address defined. If there is an IP address defined the correspondig config which on a RedHat linux box is found in */etc/sysconfig/network−scripts/ifcfg−<interface name>* will be used, else the interface is set up as IP−less in promiscuous mode.

THIS HAS NOT YET BEEN TESTED WITH ANYTHING ELSE THAN ETHERNET INTERFACES! I WILL HOPEFULLY SOON REVIEW IT WITH ISDN INTERFACES AND REPORT HOW THE DIFFERENCES ARE!

A single snort process is then started on each interface, and also *swatch* will be started to check for errors when restarting snort for rule updates (see the *swatch* section below).

When shutting down snort all IP−less interfaces will be shut down but not any interfaces with existing IP configurations because that could last to inaccessability if the "snort'ed" interface is vital for the snort sensor (learned that the hard way >;)

Maybe a better solution would be to check the interface's config file for an entry like

```
     ONBOOT=yes
```

and only if there is not *yes* then the interface will be shut down. But that's not yet implemented.

Now here is the extended snort initscript:

```
#!/bin/sh
```

```
#
# snortd         Start/Stop the snort IDS daemon.
#
# chkconfig: 2345 40 60
# description:  snort is a lightweight network intrusion detection tool that
#               currently detects more than 1100 host and network
#               vulnerabilities, portscans, backdoors, and more.
#
# June 10, 2000 −− Dave Wreski Dave Wreski <dave at linuxsecurity.com>
#   − initial version
# July 08, 2000 Dave Wreski <<dave at guardiandigital.com>
#   − added snort user/group
#   − support for 1.6.2
# April 11, 2001 Sandro Poppi <spoppi at gmx.de>
#   − added multiple interfaces option for use with dial up lines
#     or more than one sniffer interface
#     I don't think the libpcap option to use "−i any" is a good choice,
#     because snort would be set up to monitor one or more ip−less interfaces
#     while leaving the monitor interface "unprotected"
#   − changed the subsystem name from snort to snortd to get rid of error messages
#     when rebooting (the killall script on a redhat box depends on the correct name)
#   − added a function daemonMult derived from the function daemon in /etc/rc.d/init.d/functions
#     to allow starting multiple instances of snort with the convenience of the daemon function
#     (eventually this could be integrated into the normal daemon function of redhat, have to get
#     in touch with the author)
# January 01, 2002 Sandro Poppi <spoppi at gmx.de>
#   − added check if swatch is installed
#   − added check for interfaces other than ethernet since only those are expected to work with i
#
# Source function library.
. /etc/rc.d/init.d/functions

# A function to start a program even more than once
# rewritten version of the daemon function in /etc/rc.d/init.d/functions
daemonMult() {
        # Test syntax.
        gotbase=
        user=
        nicelevel=0
        while [ "$1" != "${1##−}" −o "$1" != "${1##+}" ]; do
          case $1 in
            '')    echo '$0: Usage: daemon [+/−nicelevel] {program}'
                   return 1;;
            −−check)
                   shift
                   base=$1
                   gotbase="yes"
                   shift
                   ;;
            −−user)
                   shift
                   daemon_user=$1
                   shift
                   ;;
            −*|+*) nicelevel=$1
                   shift
                   ;;
             *)    nicelevel=0
                   ;;
          esac
        done
        # Save basename.
        [ −z $gotbase ] && base=`basename $1`

        # make sure it doesn't core dump anywhere; while this could mask
        # problems with the daemon, it also closes some security problems
        ulimit −S −c 0 >/dev/null 2>&1

        # Echo daemon
        [ "$BOOTUP" = "verbose" ] && echo −n " $base"

        # And start it up.
        if [ −z "$daemon_user" ]; then
           nice −n $nicelevel initlog $INITLOG_ARGS −c "$*" && success "$base startup" || failure
        else
           nice −n $nicelevel initlog $INITLOG_ARGS −c "su $daemon_user −c \"$*\"" && success "$b
        fi
}

# Specify your network interface(s) here
INTERFACE="eth1 eth2"

# See how we were called.
case "$1" in
start)
        if [ −x /usr/bin/swatch ] ; then
          echo −n "Starting swatch: "
          # inserted poppi to make use of swatch
          # starting it before snort to get hints on startup errors of snort
          # if using the snort option −s use /var/log/secure,
          # if using output alert_syslog: in snort.conf use /var/log/messages
          /usr/bin/swatch −−daemon −−tail /var/log/messages −−config−file /etc/swatch/swatchrc &
          touch /var/lock/subsys/swatch
          echo "done."
          echo
        fi

        # added multiple interfaces option
        for i in `echo "$INTERFACE"` ; do
          echo −n "Starting snort on interface $i: "
          # inserted to implement ip−less sniffer interface for snort at startup
          # if the interface is not yet loaded or if the interface isn't up yet
          if [ `/sbin/ifconfig $i 2>&1 | /bin/grep −c "Device not found"` = "0" \
               −o `/sbin/ifconfig $i 2>&1 | /bin/grep −c "UP"` = "0" ] ; then

            # check for interfaces other than ethernet!
            if [ `echo $i | /bin/grep −c "^eth"` = "1" ] ; then
              # check if there is a config for the given interface
              # normally this should be omitted for security reasons for a sniffer interface
              if [ −s "/etc/sysconfig/network−scripts/ifcfg−$i" ]; then
                # use the config
                /sbin/ifup $i
              else
                # ip less sniffer interface
                /sbin/ifconfig $i up promisc
              fi
            fi
          fi
          # call the rewritten daemon function from above
          daemonMult /usr/sbin/snort −u snort −g snort −d −D \
                 −i $i −I −l /var/log/snort −c /etc/snort/snort.conf
          echo
        done
        touch /var/lock/subsys/snortd

        ;;
  stop)
        echo −n "Stopping snort: "
        killproc snort
        rm −f /var/lock/subsys/snortd

        # inserted Poppi
        if [ −x /usr/bin/swatch ] ; then
          echo
          echo −n "Stopping swatch: "
          kill `ps x|grep "/usr/bin/swatch"|grep −v grep|awk '{ print $1 }'`
          rm −f /var/lock/subsys/swatch
        fi

        # shutdown interface if and only if it has NO ip address
        # and if it is a ethernet interface
        # this is done because we don't want to shutdown interfaces still needed
        for i in `echo "$INTERFACES"`; do
          if [`echo $i | /bin/grep −c "^eth"` = "1" −a \
              `/sbin/ifconfig $i 2>&1 | /bin/grep −c "inet addr:"` = "0" ] ; then
            /sbin/ifconfig $i down
          fi
        done
        echo
        ;;
  restart)
        $0 stop
        $0 start
        ;;
  status)
        status snort
        #status swatch
        ;;
  *)
        echo "Usage: $0 {start|stop|restart|status}"
        exit 1
esac
exit 0
```

4.2.3. /etc/snort/snort−check

This shell script is used to generate winpopups via *smbclient* or sending emails to given persons. It was inspired by Bill Richardson's script published on the snort homepage.

The winpopup part may be obsoleted by the *smb* output module introduced in snort 1.8 but I haven't tested it yet.

```
#!/bin/sh

# Script to be run from within swatch to send alerts in multiple formats
# inspired from script on www.snort.org by Bill Richardson
# extended to read a file called "hosts" with names of
# workstation to send a winpopup, syntax is the same as with snortd option −M
# Poppi, 02.05.2001
# Prerequisites:
# Samba set up correctly
# Change the following variables according to your system (for RedHat 7.x user it should be ok)

# hostfile holds the name of the file containing the workstation for winpopups
hostfile="/etc/snort/hosts"

# recipientfile holds the addresses of all recipients in a single file,
# seperated by newline
recipientfile="/etc/snort/recipients"

# if a recipient file exists
if [ −s "$recipientfile" ] ; then
  # generate the recipientlist with email adresses.
  for i in `cat $recipientfile` ; do
    recipients="$recipients "$i
  done

  echo "$*" | mail −s "Snort−Alert!!!" "$recipients"
fi

# if a hostfile exists, send winpopups
if [ −s "$hostfile" ] ; then
  for i in `cat $hostfile` ; do
    echo "Snort−Alert! $*" | smbclient −M $i > /dev/null 2>&1
  done
fi
```

4.2.3.1. /etc/snort/hosts

In this file you put in all the workstation names of the hosts which should get the snort message, one per line:

```
       ws001
       ws002
       ws003
```

4.2.3.2. /etc/snort/recipients

In */etc/snort/recipients* you put in email addresses of recipients who wish (or are urged to ;) receive your snort alarms, one address per line:

```
       jane@internal.local.com
       henk@snort.info
       sandro@snort.info
```

If any of these two files is omitted then the corresponding feature is disabled.

4.2.4. Snort internal Statistics

Snort has the ability built in to print out some internal statistics. This can be achieved using the following command:

/bin/kill −SIGUSR1 <pid of snort>

or if you have more than one snort process running on the same machine and want to get info about all at once:

/bin/killall −USR1 snort

With either of these commands you get internal statistics in the following way in your syslog (*/var/log/messages* with RedHat):

```
Sep 29 07:51:48 ids01 snort[8000]:   ============================================================
Sep 29 07:51:48 ids01 snort[8000]: Snort analyzed 27316 out of 27316 packets,
Sep 29 07:51:48 ids01 snort[8000]: dropping 0(0.000%) packets
Sep 29 07:51:48 ids01 snort[8000]: Breakdown by protocol:                Action Stats:
Sep 29 07:51:48 ids01 snort[8000]:     TCP: 27152      (99.400%)         ALERTS: 0
Sep 29 07:51:48 ids01 snort[8000]:     UDP: 0          (0.000%)          LOGGED: 0
Sep 29 07:51:48 ids01 snort[8000]:    ICMP: 164        (0.600%)          PASSED: 0
Sep 29 07:51:48 ids01 snort[8000]:     ARP: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]:    IPv6: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]:     IPX: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]:   OTHER: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]: DISCARD: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]: ==============================================================
Sep 29 07:51:48 ids01 snort[8000]: Fragmentation Stats:
Sep 29 07:51:48 ids01 snort[8000]: Fragmented IP Packets: 0          (0.000%)
Sep 29 07:51:48 ids01 snort[8000]:     Fragment Trackers: 0
Sep 29 07:51:48 ids01 snort[8000]:    Rebuilt IP Packets: 0
Sep 29 07:51:48 ids01 snort[8000]:    Frag elements used: 0
Sep 29 07:51:48 ids01 snort[8000]: Discarded(incomplete): 0
Sep 29 07:51:48 ids01 snort[8000]:    Discarded(timeout): 0
Sep 29 07:51:48 ids01 snort[8000]:   Frag2 memory faults: 0
Sep 29 07:51:48 ids01 snort[8000]: ==============================================================
Sep 29 07:51:48 ids01 snort[8000]: TCP Stream Reassembly Stats:
Sep 29 07:51:48 ids01 snort[8000]:         TCP Packets Used: 27152      (99.400%)
Sep 29 07:51:48 ids01 snort[8000]:          Stream Trackers: 1
Sep 29 07:51:48 ids01 snort[8000]:           Stream flushes: 0
Sep 29 07:51:48 ids01 snort[8000]:            Segments used: 0
Sep 29 07:51:48 ids01 snort[8000]:    Stream4 Memory Faults: 0
Sep 29 07:51:48 ids01 snort[8000]: ==============================================================
```

But remember: With versions prior to 1.8.3 you have to restart snort to get new statistics, so always combine the **kill −SIGUSR1** with a snort restart if not using the actual version!

You first should have a look on the first 2 lines. If snort tells you that there are dropped packets you have to take a very close look on your configuration of the snort box itself not only (but including) the snort configuration.

E.g. stop all unnecessary services which are not vital for the box. And take a look on the output of the **top** command. If the idle counter is very low you should figure out which processes eat up all of your cpu time and eventually outsource the corresponding program packets. This is e.g. true when using ACID and the underlying database and snort on the same machine with less memory and/or cpu.

The other statistical data lines give you an overview of some of the preprocessors and their work. You should also have a look on the memory faults sections. If the number is not 0 you should have a look on your memory usage and eventually configure the preprocessors to use more memory (take a look to the appropriate section in */etc/snort/snort.conf*).

Now a short script which I was inspired by Greg Sarsons to get snort's internal statistics, save them to a file and restart snort.

The statistics file will be archived to */var/log/snort/archive* so you have to create that directory first ;)

```
#!/bin/bash
# Script to generate and extract snort statistics from syslog or given file
# generated after kill −USR1 <snort−pid>
#
# This script assumes that the pid is logged into the logfile!
# This can be obtained using  the following line in snort.conf:
# output alert_syslog: LOG_AUTH LOG_ALERT LOG_PID
#
# (c) Sandro Poppi 2001
# Released under GPL

echo "Starting gathering snort internal statistics. Please be patient..."

if [ "$1." == "." −o ! −e "$1" ] ; then
  # no or unexistent file given, using default
  log_file="/var/log/messages"

else
  # when using non−standard logfile location make sure snort uses this logfile
  # when sending signal USR1 else this script won't work!
  log_file="$1"
fi

# find out snort pids
snort_pid=`/sbin/pidof snort`

# get internal statistics for all snort processes
# not using killall to get already sorted output
for i in `echo $snort_pid` ; do
  kill −USR1 $i

  # sleep for 2 secs to let snort time to send statistics to syslog ;)
  sleep 2
done

# immediately restart snort after sending signal USR1
# this may be ommitted when using CVS version of snort after about 01.11.2001
# or any version from 1.8.2 or higher
/etc/rc.d/init.d/snortd restart

for i in `echo $snort_pid` ; do
  # process logfile

  filename=/var/log/snort/archive/snort.`date "+%Y−%m−%d"`.$i.log

  # check for existing file and rename it if existing
  if [ −e "$filename" ] ; then
```

```
    mv "$filename" "$filename.bak"
  fi

  egrep "snort\[$i\]:" $log_file > "$filename"

  # check if there are dropped packets using lines like
  # Oct 22 18:02:06 xbgh17183 snort[573]: dropping 0(0.000%) packets
  if [ "`egrep "dropping" $filename | awk -F "[ (]" '{ print $7 }'`" != "0" -a \
       "`egrep -c "dropping" $filename`" != "0" ] ; then
    echo "Snort's dropping packets!!! Take a look on the configuration and/or the system's perfor
  fi

done

echo "Gathering snort internal statistics finished..."
```

4.2.5. Testing Snort

To test snort you should edit */etc/rc.d/init.d/snortd* and make the interface listen on the loopback device *lo*. For people with a network card installed you can use *eth0* instead but you have to use a second pc to run snot because no packet is sent over the interface if snot and snort are run on the same machine!

Probably the simplest way to test snort is to use *snot* which can be found on http://www.sec33.com/sniph/.

You have to have libnet installed for snot. Since on RedHat 7.x there is no RPM available you could use *libnet−1.0.2−6mdk.i586.rpm* from Mandrake Soft, which can be found on http://rpmfind.net/ and of course on Mandrake's site http://www.mandrake.com/. Most Mandrake RPMs could be used with no problem on a RedHat system. But be warned: Mandrake does not provide *i386* RPMs so you can't use them with a processor less than an old Pentium P5. In such a case you have to get the sources from http://www.packetfactory.net/projects/libnet and compile it from scratch yourself.

To compile snot you only have to untar the tarball, cd into the snot directory and call *make*. If compilation exits without an error snot is ready to use, if not you are almost always missing some development packages.

To prepare snot you should first copy */etc/snort/snort.conf* into the snot directory and *cat* one or more rule files to the end of the copied *snort.conf* using e.g.:

cat /etc/snort/backdoor.rules >> **snort.conf**

Then on one console you should call **tail −f /var/log/messages**, while on another you should try to run the tests.

Snot can then be called the following way assuming you used *lo* as the interface name in the snortd initscript:

./snot −r snort.conf −d localhost −n 5

With that command you tell snot to use the copied *snort.conf*, the destination is *localhost* and for not triggering too many alerts restrict it to a maximum of 5.

You'll probably get some messages saying ignoring additional parameters because snot can not handle yet the new parameters introduced in snort 1.8. Don't panic, just ignore the messages, snot works fine though.

In */var/log/messages* you should now see some snort alerts, e.g.:

```
Sep 10 18:22:33 ids01 snort[1536]: <lo> GateCrasher access: 192.168.213.151:6969 -> 127.0.0.1:317
Sep 10 18:22:33 ids01 snort[1536]: <lo> GateCrasher access: 192.168.213.151:6969 -> 127.0.0.1:317
Sep 10 18:22:33 ids01 snort[1536]: <lo> GateCrasher access: 192.168.155.231:6969 -> 127.0.0.1:575
Sep 10 18:22:33 ids01 snort[1536]: <lo> GateCrasher access: 192.168.155.231:6969 -> 127.0.0.1:575
Sep 10 18:22:33 ids01 snort[1536]: <lo> Deep Throat access: 192.168.170.42:2140 -> 127.0.0.1:6052
```

If you get similiar alerts it's ok, if not please take again a look on your configuration until you get this far.

Now it's time to edit */etc/snort/snort.conf* again and put in the correct value to the *INTERFACE* variable, restart snort and get a cup of coffee. You have deserved it!

4.3. Configuring MySQL

To allow Snort to send alerts to MySQL you first have to install MySQL. With most linux distributions there are MySQL packages available so you should use them. If not you'll probably have to compile and install it from scratch by downloading the tarball from http://www.mysql.org/. Take a look at the documentation shipped with MySQL to set it up.

When you have a running MySQL daemon (with RedHat after installing the RPMs run **/etc/rc.d/init.d/mysql start**) you have to initialize a snort database. This is documented in the next section.

Since there should be a password set for each account you'll have to use the *−p* option on the mysql commandline.

```
[root@ids01 /root]# mysql −u root −p
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with −A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 133 to server version: 3.23.32

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql>create database snort;
Query OK, 1 row affected (0.00 sec)

mysql> connect snort
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with −A

Connection id:    139
Current database: snort

mysql> status
−−−−−−−−−−−−−−
mysql  Ver 11.12 Distrib 3.23.32, for redhat−linux−gnu (i386)

Connection id:          139
Current database:       snort
Current user:           root@localhost
Current pager:          stdout
Using outfile:          ''
```

```
Server version:         3.23.32
Protocol version:       10
Connection:             Localhost via UNIX socket
Client characterset:    latin1
Server characterset:    latin1
UNIX socket:            /var/lib/mysql/mysql.sock
Uptime:                 1 day 2 hours 6 min 21 sec

Threads: 14  Questions: 4272  Slow queries: 0  Opens: 58  Flush tables: 1  Open tables: 18 Querie
−−−−−−−−−−−−−−

mysql> grant CREATE,INSERT,SELECT,DELETE,UPDATE on snort.* to snort@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
```

To generate the required table structure of the database use the *create_mysql* script which can be found in the contrib section of the original tarball or my RPM.

[root@ids01 /root]# mysql −u root −p snort < ./contrib/create_mysql

You'll have to add a userid/password pair for the database, remember to change *xxxx* to a password suitable for your environment!

```
[root@ids01 /root]# mysql −u root −p mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with −A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 148 to server version: 3.23.32

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> insert into user (User,Password) values('snort',PASSWORD('xxxx'));
Query OK, 1 row affected (0.00 sec)

mysql> exit
Bye
```

Now add some extra tables for your convenience shipped in the contrib section of the snort tarball and my RPM using the command

zcat snortdb−extra.gz | mysql −u root −p snort

If you wish to use the archiving feature of ACID you'll have to create another database *snort_archive* (or any other name you prefer) exactly the same way as you defined the *snort* database.

From now on the database is ready to be used for logging with the database output module of snort which you could now activate in */etc/snort/snort.conf*.

4.4. Configuring ADODB

ADODB is a required part for ACID. It delivers database connection support for PHP based programs like ACID.

Install ADODB in a directory available for your webserver. On a RedHat box this usually is */var/www/html/adodb/*.

In ADODB version 1.31 there is a bug in *adodb.inc.php* which may still exist in newer versions. You'll have to change the path in line 40 to reflect your local requirements. It's vital to delete the command **dirname()** completely so that it looks like this:

```
if (!defined('_ADODB_LAYER')) {
       define('_ADODB_LAYER',1);

       define('ADODB_FETCH_DEFAULT',0);
       define('ADODB_FETCH_NUM',1);
       define('ADODB_FETCH_ASSOC',2);
       define('ADODB_FETCH_BOTH',3);

       GLOBAL
               $ADODB_vers,            // database version
               $ADODB_Database,        // last database driver used
               $ADODB_COUNTRECS,       // count number of records returned − slows down query
               $ADODB_CACHE_DIR,       // directory to cache recordsets
               $ADODB_FETCH_MODE;      // DEFAULT, NUM, ASSOC or BOTH. Default follows native dr

       $ADODB_FETCH_MODE = ADODB_FETCH_DEFAULT;
       /**
        * SET THE VALUE BELOW TO THE DIRECTORY WHERE THIS FILE RESIDES
        * ADODB_RootPath has been renamed ADODB_DIR
        */
       if (!defined('ADODB_DIR')) define('ADODB_DIR','/var/www/html/adodb');
```

That's all what has to be done with ADODB.

4.5. Configuring PHPlot

After downloading PHPlot just tar the package into a directory visible for your webserver. On a RedHat box this usually is */var/www/html/phplot/*. Nothing to configure here.

4.6. Configuring ACID

As stated before ACID needs a couple of additional programs installed to work correctly. While a database system like MySQL version 3.23+, a webserver with PHP 4.0.2+ support like *apache* with the PHP module *mod_php* and ADODB version 0.93+ are required, the graphics library *gd* version 1.8+ and PHPlot version 4.4.6+ are optional but recommended. Since *apache*, the PHP module and *gd* are almost always included and installed with any linux distribution they are not covered in this document.

For snort 1.8+ you'll need at least ACID 0.9.6b13. ACID is shipped with my RPM in the contrib section but may be an outdated version since ACID is developed rapidly. So you should always have a look at ACID's homepage if a newer version exists.

Install ACID into a directory visible to your webserver like */var/www/html/acid/*.

In */var/www/html/acid/acid_conf.php* you'll have to edit some variables to suit your environment.

First of all define the database type in the variable *DBtype*. Next define all *alert_** and *archive_** variables.

In *ChartLib_path* you define the path to PHPlot, in our case */var/www.html/phplot*.

The last variable you have to define is *portscan_file* where you put in the complete path and filename of snort's portscan logfile.

All other variables should be sufficient for now. You can edit them to suit your needs.

Here's the config I use:

```
<?php

$ACID_VERSION = "0.9.6b15";

/* Path to the DB abstraction library
 *  (Note: DO NOT include a trailing backslash after the directory)
 *   e.g. $foo = "/tmp"      [OK]
 *        $foo = "/tmp/"     [OK]
 *        $foo = "c:\tmp"    [OK]
 *        $foo = "c:\tmp\"   [WRONG]
 */
$DBlib_path = "/var/www/html/adodb";

/* The type of underlying alert database
 *
 *  MySQL       : "mysql"
 *  PostgresSQL : "postgres"
 */
$DBtype = "mysql";

/* Alert DB connection parameters
 *   − $alert_dbname   : MySQL database name of Snort alert DB
 *   − $alert_host     : host on which the DB is stored
 *   − $alert_port     : port on which to access the DB
 *   − $alert_user     : login to the database with this user
 *   − $alert_password : password of the DB user
 *
 *  This information can be gleaned from the Snort database
 *  output plugin configuration.
 */
$alert_dbname   = "snort";
$alert_host     = "localhost";
$alert_port     = "";
$alert_user     = "snort";
$alert_password = "xxxx";

/* Archive DB connection parameters */
$archive_dbname   = "snort_archive";
$archive_host     = "localhost";
```

```
$archive_port     = "";
$archive_user     = "snort";
$archive_password = "xxxx";

/* Type of DB connection to use
 *   1  : use a persistant connection (pconnect)
 *   2  : use a normal connection (connect)
 */
$db_connect_method = 1;

/* Path to the graphing library
 *  (Note: DO NOT include a trailing backslash after the directory)
 */
$ChartLib_path = "/var/www/html/phplot";

/* File format of charts ('png', 'jpeg', 'gif') */
$chart_file_format = "png";

/* Chart default colors − (red, green, blue)
 *    − $chart_bg_color_default    : background color of chart
 *    − $chart_lgrid_color_default : gridline color of chart
 *    − $chart_bar_color_default   : bar/line color of chart
 */
$chart_bg_color_default     = array(255,255,255);
$chart_lgrid_color_default  = array(205,205,205);
$chart_bar_color_default    = array(190, 5, 5);

/* Maximum number of rows per criteria element */
$MAX_ROWS = 20;

/* Number of rows to display for any query results */
$show_rows = 50;

/* Number of items to return during a snapshot
 *  Last _X_ # of alerts/unique alerts/ports/IP
 */
$last_num_alerts = 15;
$last_num_ualerts = 15;
$last_num_uports = 15;
$last_num_uaddr = 15;

/* Number of items to return during a snapshot
 *  Most Frequent unique alerts/IPs/ports
 */
$freq_num_alerts = 5;
$freq_num_uaddr = 15;
$freq_num_uports = 15;

/* Number of scroll buttons to use when displaying query results */
$max_scroll_buttons = 12;

/* Debug mode     − how much debugging information should be shown
 * Timing mode    − display timing information
 * SQL trace mode − log SQL statements
 *   0 : no extra information
 *   1 : debugging information
 *   2 : extended debugging information
 *
 * HTML no cache − whether a no−cache directive should be sent
 *                 to the browser (should be = 1 for IE)
 *
 * SQL trace file − file to log SQL traces
 */
$debug_mode = 0;
$debug_time_mode = 1;
$html_no_cache = 1;
$sql_trace_mode = 0;
$sql_trace_file = "";

/* Auto−Screen refresh
 * − Refresh_Stat_Page − Should certain statistics pages refresh?
 * − Stat_Page_Refresh_Time − refresh interval (in seconds)
 */
$refresh_stat_page = 1;
$stat_page_refresh_time = 180;

/* Display First/Previous/Last timestamps for alerts or
 * just First/Last on the Unique Alert listing.
 *    1: yes
 *    0: no
 */
$show_previous_alert = 1;

/* Sets maximum execution time (in seconds) of any particular page.
 * Note: this overrides the PHP configuration file variable
 *       max_execution_time.  Thus script can run for a total of
 *       ($max_script_runtime + max_execution_time) seconds
 */
$max_script_runtime = 180;

/* How should the IP address criteria be entered in the Search screen?
 *   1 : each octet is a separate field
 *   2 : entire address is as a single field
 */
$ip_address_input = 2;

/* Resolve IP to FQDN (on certain queries?)
 *    1 : yes
 *    0 : no
 */
$resolve_IP = 0;

/* Should summary stats be calculated on every Query Results page
 * (Enabling this option will slow page loading time)
 */
$show_summary_stats = 1;

/* DNS cache lifetime (in minutes) */
$dns_cache_lifetime = 20160;

/* Whois information cache lifetime (in minutes) */
$whois_cache_lifetime = 40320;

/* Snort spp_portscan log file */
$portscan_file = "/var/log/snort/portscan.log";

/* Event cache Auto−update
 *
 *  Should the event cache be verified and updated on every
 *  page log?  Otherwise, the cache will have to be explicitly
 *  updated from the 'cache and status' page.
 *
 *  Note: enabling this option could substantially slow down
 *  the page loading time when there are many uncached alerts.
 *  However, this is only a one−time penalty.
 *
 *   1 : yes
 *   0 : no
 */
$event_cache_auto_update = 1;

/* Link to external Whois query */
$external_whois_link = "http://www.samspade.org/t/ipwhois?a=";

?>
```

You wonder why I use *xxxx* as password? Well, do you like your password to be available for everyone in the world? j/k >8)

When first calling ACID via your browser you'll get a hint that you have to install ACID support in the chosen database. Click on *Setup* and ACID should create the required entries in the database. If everything is set up correctly you'll get all informations which are currently in the database, normally nothing at this time ;)

Try to trigger some snort rules with *snot* (see section above) or e.g. *nmap* (see http://www.nmap.org/, a portscanner with many more capabilities) or *nessus* (see http://www.nessus.org/, a security scanner to find vulnerabilities of a system).

Now you should get all alarms right the time they happen with ACID.

4.7. Configuring SnortSnarf

SnortSnarf is another tool which analyses snort's logfile instead of a database.

Install SnortSnarf by taring it into a directory you like, I use */opt/SnortSnarf/*.

Copy */opt/SnortSnarf/Time−modules/lib/Time* to */opt/SnortSnarf/include/SnortSnarf/Time* to make the required perl modules available for SnortSnarf .

Copy the following files to the webserver's *cgi−bin* directory (e.g. */var/www.cgi−bin/*):

```
     /opt/SnortSnarf/cgi/*
     /opt/SnortSnarf/include/ann_xml.pl
     /opt/SnortSnarf/include/web_utils.pl
     /opt/SnortSnarf/include/xml_help.pl
```

If you would like to use the annotation feature with which you can create notes to an incident in SnortSnarf you first have to create the directory */var/www/html/SnortSnarf/annotations*, copy */opt/SnortSnarf/new−annotation−base.xml* to */var/www/html/SnortSnarf/annotations* and call

./setup_anns_dir.pl −g apache /var/www/html/SnortSnarf/annotations

in */opt/SnortSnarf/utilities*.

Check the rights in */var/www/html/SnortSnarf/annotations* and make them look like this:

```
[root@ids01 SnortSnarf]# ll −a /var/www/html/SnortSnarf/annotations/
total 16
drwxrwx−−−    2 root     apache       4096 May 23 14:31 .
drwxr−xr−x    8 root     root         4096 May 23 14:17 ..
−rw−r−−r−−    1 apache   apache        478 May 23 14:31 new−annotation−base.xml
```

I created a wrapper script called *opt/SnortSnarf/snortsnarf.sh* to get rid of the nasty @INC errors (someone with better perl know−how could give me a hint how to get rid of the errors, thx). I'm calling */opt/SnortSnarf/snortsnarf.sh* via cron every hour from 6 am to 6 pm.

My crontab enrty looks like this:

```
# generate SnortSnarf statistics every hour from 6am to 6pm
0 6,7,8,9,10,11,12,13,14,15,16,17,18 * * * /opt/SnortSnarf/snortsnarf.sh
```

SnortSnarf is called to analyse five logfiles */var/log/messages**, put the generated HTML files into */var/www/html/SnortSnarf* and make use of the annotation feature which is described above.

Here's the */opt/SnortSnarf/snortsnarf.sh* listing:

```
#!/bin/sh
# wrapper for use with crontab to get rid of the @INC problem
# Poppi, 22.05.2001
cd /opt/SnortSnarf
./snortsnarf.pl −d /var/www/html/SnortSnarf −db /var/www/html/SnortSnarf/annotations/new−annotati
```

Test SnortSnarf by calling *snortsnarf.sh* and take a look with your browser to */var/www/html/SnortSnarf/*.

4.8. Configuring Arachnids_upd

Be warned: Automatic updating the rules without any encryption or athentication can create backdoors because the rules could be compromised to allow an attacker to be hidden from your IDS! So use that with care!

Another issue is that www.whitehats.com is often offline so no rules can be downloaded.

Untar the arachnids_upd package to a directory of your choice, I choose */opt/arachnids_upd/*.

For snort 1.8+ you'll have to edit */opt/arachnids_upd/arachnids_upd.pl* and change the filename of the file to download to:

```
     my $url = "http://www.whitehats.com/ids/vision18.rules.gz";   # Default URL.
```

Since Arachnids_upd makes use of *wget* it should be installed on your system and configured to work with your internet connection.

An example version of ~.wgetrc is shown here for connecting via a proxy server with user authentication:

```
    proxy_user = user
    proxy_passwd = xxxx
    http_proxy = <proxy>:<port>
    ftp_proxy = <proxy>:<port>
    use_proxy = on
```

Replace <proxy> with the name or ip address of your proxy and <port> with the port number the proxy uses. If you don't use a proxy you don't need any of these entries.

Again I created a shell script to get new rules, change the variable names of *vision.rules* to suite the definition in */etc/snort/snort.conf* and restart snort for the new rules to take effect.

```
#!/bin/sh
# Script to generate the correct updates of vision.rules using arachnids_upd.pl
# Poppi 22.05.2001

# get new rules (requires ~/.wgetrc to be set up to access internet)
/opt/arachnids_upd/arachnids_upd.pl −o /opt/arachnids_upd/vision.rules −b /opt/arachnids_upd/rule

# change the variable names according to the ones used in /etc/snort/snort.conf and copy the new
cat /opt/arachnids_upd/vision.rules | sed s/EXTERNAL/EXTERNAL_NET/g | sed s/INTERNAL/HOME_NET/g >

# restart snort for the rules to take effect
/etc/rc.d/init.d/snortd restart
```

As arachnids_upd is also capable of deleting rules in *vision.rules* while downloading you can if you like edit */opt/arachnids_upd/arachnids.ignore* and put in the IDS numbers which should be ignored.

```
    # Put the IDS numbers of the rules that should be disabled in here.
    # One number per line.

    # Examples:

    1      # Ignore IDS1
    2      # Ignore IDS2
    3      # Ignore ISD3

    # I think you get it now :)
```

4.9. Configuring Swatch

Swatch is an excellent package to take care for any logfile. It can be configured using regular expressions to alert if anything bad is logged in the logfile.

Swatch requires the following perl modules to be installed:

```
    perl−TimeDate
    perl−Date−Calc
    perl−Time−HiRes
    perl−File−Tail
```

Swatch is available as an RPM from http://www.lug−burghausen.org/projects/Snort−Statistics/swatch−3.0.2−1.noarch.rpm along with the source RPM I created http://www.lug−burghausen.org/projects/Snort−Statistics/swatch−3.0.2−1.src.rpm.

Swatch is configured via a single config file */etc/swatch/swatch.conf*.

I'm shipping it with a demo *swatch.conf* containing two rules for snort messages and snort errors shown below along with some other examples from the original swatch package.

```
# global swatch.conf file
# * Poppi, 30.04.2001
# − initial version
#
# * Poppi, 08.06.2001
# − added error support; make sure to start swatch BEFORE snort ;)
#
# Poppi, 19.09.2001
# − added throttle for not getting too much alarms of the same incident

# normal snort messages (with PID)
# get rid of double alerts for 10 secs, e.g. pings
watchfor /snort\[/
        bell
        exec /etc/snort/snort-check $0
        throttle 00:00:10

# snort error messages could be with or without the [!] indicator
watchfor /snort: (\[\!\])* ERROR/
        bell
        exec /etc/snort/snort-check $0
```

The first rule is for getting all alerts generated via the output module *alert_syslog*, the second for getting any error messages snort generates at startup if anything went wrong (like errors in a rule file).

Both rules do ring the pc bell (well, if the sensor is used in a room without operators in sight this does not make much sense ;) and make use of the *snort−check* script described before to alert the given persons. In *$0* swatch gives you the complete line of the logfile entry which triggered swatch.

Swatch has to be started prior to snort. Instead of generating an own swatch initscript with the correct *chkconfig* dates I chose to include it in */etc/rc.d/init.d/snortd* because the dependencies of my use of swatch are such that I − again for me − decided to do that. I know that's not the "fine english way", and the swatch part can be put into an own initscript relatively easy. Maybe I will change this in the future.

5. Security Issues

Snort is running under an own userid/group pair *snort/snort*. This should make sure that any buffer overflow not yet fixed (if any) only gets the rights the snort user has. For people for whom this is not enough you might use a changeroot'ed environment using snort's command line option *−t*. But please don't ask me how to create it, I've never done it and maybe will not do it anytime.

As with all security related systems don't allow more services as needed. If you do a standard installation of any linux distribution take a look into */etc/inetd.conf* if your distribution is still using the older inetd or */etc/xinetd.d/** on an *xinetd* based system and disable all services not really vital for your system. E.g. you don't want to use telnet, replace it with ssh.

Also take a look at the initscripts, on a Sytem V based system like RedHat found in */etc/rc.d/init.d/**. If there are any services like *nfs* and *portmap* which you don't use on such a system delete the corresponding packages completely.

And you should read a lot of security related papers and HOWTOs, like the *Security−HOWTO*, the *System Administrators Guide* or *Network Administrator guide.*

Or take a look on various security related websites like http://www.securityfocus.com/, http://www.linuxsecurity.org/ or http://www.insecure.org/

6. Getting Help

In the end you might find yourself unable to solve your problems and need help from someone else. The most efficient way is either to ask someone local or in your nearest Linux user group, search the web for the nearest one.

But first of all try a look on http://www.snort.org/ and the snort mailinglists. The people out there helped me very much.

Another possibility is to ask on Usenet News in one of the many, many newsgroups available. The problem is that these have such a high volume and noise (called low signal−to−noise ratio) that your question can easily fall through unanswered.

No matter where you ask it is important to ask well or you will not be taken seriously. Saying just *snort does not work* is not going to help you and instead the noise level is increased even further and if you are lucky someone will ask you to clarify.

Instead describe your problems in some detail that will enable people to help you. The problem could lie somewhere you did not expect. Therefore you are advised to list the following information about your system:

Software

- ♦ /etc/snort/snort.conf
- ♦ /etc/swatch/swatch.conf if used
- ♦ excerpt of /var/log/messages, but only filter the relevant entries
- ♦ used Linux distribution or operating system and version
- ♦ Software that shows the error (with version number or date)

And you can ask me directly. But please remember: I'm having a live beyond computers and my spare time is rare. I will almost always answer my emails but this can take some times. Also I'm subscribed to the snort−users mailinglist too so you reach me this way too.

7. Questions and Answers

This is just a collection of what I believe are the most common questions people might have. Give me more feedback and I will turn this section into a proper FAQ.

- Q:

  A: