

## Complete Snort-based IDS Architecture, Part Two

*Anton Chuvakin and Vladislav V. Myasnyankin* 2002-11-19

### Complete Snort-based IDS Architecture, Part Two

by [Anton Chuvakin](#), Ph.D. and Vladislav V. Myasnyankin

last updated November 19, 2002

---

Many companies find it hard to justify acquiring the IDS systems due to their perceived high cost of ownership. However, not all IDS systems are prohibitively expensive. This is second part of a two-part article that will provide a set of detailed directions to build an affordable intrusion detection architecture from hardware and freely available software. In this installment we shall discuss Web interface configuration, summaries and daily reporting, automated attack response, sensor installation, installation of the central station, and big distributed IDS systems.

#### Web Interface Configuration

First, you should deploy an Apache Web server with an SSL support, if it is not already installed by the Debian config. The command to run is "apt-get install apache-ssl". When configuring, you will be asked for some information that is required to generate the SSL key pair. One need to enter the same server name as was used for the base Linux set-up.

Next, ACID IDS console is deployed via "apt-get install acidlab". This is yet another point when the choice of Debian becomes clear, as there are no packages to compile and no dependences to troubleshoot. Answering the set-up questions is easy. The only one that needs special attention is the question about the database user: it is not root as suggested by the set-up script, but the "acid" user, which we already created during the database set-up. You should also agree to inserting string with PHP module into config file and running the apache config script.

At the time of writing, there was a small bug in the ACID package install script: by default, the script does not install MySQL support for PHP needed for ACID. So, this should be done by hand:

```
# apt-get install php4-mysql
```

and Apache should be restarted as follows for the changes to take effect:

```
/etc/init.d/apache-ssl restart
```

Now, the system can be tested. For this one should go to the appropriate page using HTTPS protocol: `https://<the server address or name>/acidlab/` . On the first load, one will be asked to click on the "Setup Page" link to complete the installation. On this page the "create\_AG" button should be pushed. With this, set-up is almost complete; the only remaining part is to limit the access to the server via Apache basic authentication.

The following lines should be added to the `/etc/acidlab/apache.conf` file after the "AllowOverride None" line:

```
AuthType Basic
AuthName "Restricted"
AuthUserFile .htpasswd
Require valid-user
```

In addition, one can restrict access only from specified IP addresses. For example, if one wants to allow access to ACID console only from 192.168.2.1, 192.168.2.2 and entire 192.168.1.0 C-class network, the appropriate changes are:

```
order deny,allow
deny from all
allow from 192.168.1.0/255.255.255.0, 192.168.2.1, 192.168.2.2
```

To complete the access control set-up, one should go to the /etc/apache-ssl directory and create the password file:

```
# htpasswd -c .htpasswd <username>
```

You will be prompted for password. Several users can be added by the "htpasswd .htpasswd <username1>", etc commands.

The IDS system is now fully operational with Web access to alerts and packet data.

Some other free consoles exist for Snort. One of the better known free ones is [SnortCenter](#). This is a Web-based client-server management system written in PHP and Perl. It includes SSL-encryption, built-in user authentication, rules management and multi-language support.

The latest and greatest Snort front end is made by [Sourcefire](#), home of Marty Roesch and Snort. The slick web GUI seamlessly integrates alarm viewing with rule management, a big advantage over other Web front ends. It also provides a simple, but flexible interface for rule editing and many useful alarm viewing modes (including graphing) as well as full control over other aspects of Snort behavior, such as preprocessor configuration. Sourcefire GUI also has an option of issuing live signature updates directly from the Sourcefire site.

## Additional Features

There are some additional features, which you can use to make IDS administration and event analysis process even easier. One good idea is to add daily reporting and some attack response capabilities.

## Summaries and Daily Reporting

The most essential part of IDS deployment is monitoring of routine network activity. An effective way to accomplish this is to get daily reports on the activity. To provide daily statistics, one can query the alert MySQL database or configure Snort to also output data to syslog for summarization. The former approach can be implemented using SnortReport, which is available at <http://www.circuitsmaximus.com/download.html>. It can be used for real-time or historical reporting from the MySQL or PostgreSQL database of alarms generated by Snort.

Many tools are written to utilize the latter approach of summarizing Snort alarms from syslog. [SnortSnarf](#) by Silicon Defense is perhaps the most well known of these. It can produce HTML reports from snort alert files, include port scan summary, alert summary by alarm, alert summary by source and destination and others. Other scripts include [snort\\_stat](#) and [LogHog](#).

To simplify the analysis of events from many sensors running on the same machine (such as for the VLAN case), one can use the simple syslog filtering trick. Add a line to the corresponding `snort.conf.ethX` file:

```
output alert_syslog: LOG_LOCALX LOG_ALERT
```

In this example one uses `LOG_LOCAL1` facility for the first sensor running on the interface `eth1`. It is convenient to use different syslog facilities (`LOG_LOCAL1`, `LOG_LOCAL2` etc) to distinguish the information flows and direct them into different log files for summary reporting. For example, for a 3-sensor machine one needs to add to `/etc/syslog.conf` the following lines:

```
LOG_LOCAL1.* /var/log/snorty/snorty-eth1.log
LOG_LOCAL2.* /var/log/snorty/snorty-eth2.log
LOG_LOCAL3.* /var/log/snorty/snorty-eth3.log
```

Now, before the log rotation procedure (enabled by default on Debian) starts, you need to simply run whichever Snort script against Snort log files and mail output to the appropriate address. For example, there is `/etc/logrotate.d/syslog` entry for using "snort\_stat.pl" for daily summaries:

```
/var/log/snort/snorty-eth1.log
{
    prerotate
        /bin/cat /var/log/snort/snorty-eth1.log
    | /usr/local/bin/snort_stat.pl -i ETH1 -f | /usr/bin/mail\
cert@our.org -s "Snort Daily Report for sensor eth1"
    endscrip
    rotate 7
    create 644 root root
    daily
    compress
    postrotate
        /etc/init.d/syslog restart >/dev/null 2&>1
    endscrip
}
```

The Snort log for a sensor running on `eth1` is processed via `snort_stat.pl`, e-mailed to "cert@our.org", compressed and stored. Snort is then restarted.

## Automated Attack Response

Network intrusion detection systems are also capable of mitigating attacks by applying the network access controls, such as firewall rules, or by directly terminating offending connections. For a review of Snort attack response methods see LinuxSecurity.com's feature [Intrusion Detection Response](#).

The most flexible way to use Snort for automatic attack response is with [Guardian Active Response for Snort](#). This tool automatically updates firewall rules based on alerts generated by Snort. The new firewall rules block all incoming data from the IP address of the machine that caused Snort to generate an alert. There is a whitelist support to prevent the blocking of important machines, such as DNS servers, gateways and other predefined hosts.

The whitelist of addresses that should never be blocked by the IDS is crucial for the automated response system. Even if the IDS will invoke the response command upon seeing the attack in the full TCP stream only (thus preventing the denial of service through blocking the access to crucial hosts caused by spoofed "attack" packets), many IDS signatures have false positives that will result in disrupting of non-malicious connections. It is highly desirable to enable response only on reliable signature with no known false positives and additionally have the well-designed whitelist.

The tool supports Linux iptables, ipchains, ipfwadm, ipfw, and also commercial CheckPoint Firewall-1 and Cisco PIX firewalls.

## Distributed IDS Setup

Networks with higher traffic load might justify deploying several dedicated sensor machines reporting to a single alarm database and console. To build a distributed IDS, one needs to install the components described above into different machines. The set-up is similar to the single host set-up described above. The difference is that now you need two (or more) base machines with Debian as described above.

## Sensor Installation

Install base system, as described above. Set up Snort IDS with MySQL support:

```
# apt-get install snort-mysql
```

While configuring, accept all default values. The difference in Snort configuration is in the output plugin configuration. Now Snort is set to log to a database on another host. The appropriate lines in snort.conf on every sensor machine are:

```
# output plugins output
database: log, mysql, user=snort password=<the secret snort password
here> dbname=snort_log host=<central-log-host-running-sql-name>
```

It is apparent that every machine can have more than one Snort instance for the case of a distributed VLAN or a combination of LANs and VLANs.

## Installation of Central Station

Install base system, as described above. Set up MySQL server via "apt-get install mysql-server". During the installation, you should allow network connections to the SQL server. The following set-up matches that of a single host solution.

The differences start when granting permissions to access the database tables. One needs to repeat commands "grant" and "set password" for each host one want to connect to central database. For example:

```
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort_log.* to snort@<sensor1-
name.our.org>;
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort_log.* to snort@<sensor2-
```

```
name.our.org>;  
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort_log.* to snort@<sensor3-  
name.our.org>;
```

Installing the Web interface (including Apache and ACID set up) is similar to the previous procedure for a single host set-up. It is highly recommended to synchronize the time between all sensors and the database system. It will save you some frustration during the incident response. While running additional services (such as NTP, which has a security history) might not be desired, the sensor boxes can be set to poll a specified time server via the "rdate" command set in crontab. For example:

```
* 1 * * * /bin/rdate -s time.nist.gov
```

The multi-machine setup also brings up the problem of maintaining signature sets. It is likely that various machines will be deployed in different environments with different security requirements that will lead to corresponding differences in rulesets. It is apparent that an internal sensor will need a different rule set from the one running outside the border router or even in the DMZ.

Manual rule updates become troublesome if more than one sensor is deployed. The simple solution will be to deploy the signatures on a central server and then retrieve them via ftp ("wget" is great for this). If a signature sets are known in advance, i.e. the same signatures need to be disabled on each update, one can use "oinkmaster" from [Snort](#) to distribute updates from a central server and handle the local ruleset customization on each sensor via the oinkmaster interface.

Each sensor may also have its own unique configuration, which is tuned using "include" directive in configuration file.

The last issue to address is the security of the sensor-console communication. If deemed necessary, the SQL communication between the sensor and the database can be tunneled over the secure shell (SSH). This [article](#) provides a great reference for doing this.

## Big Distributed IDS

In this case the central system is further separated, and the Web server and SQL server are installed on separate machines. This is just a version of the previous set-up. One needs to install sensors, then install the database on one machine, and deploy a Web server and ACID onto another machine. When setting up the IDS console, one should configure the source database as located at another machine. This will leave more CPU cycles for the database process.

## Conclusions

The guide offered in this two-part series should allow readers to set-up an IDS for a small network (single host set-up) or even for a large environment (distributed configuration) at zero acquisition cost. Admittedly, there might be some set-up and maintenance costs involved, but it is apparent that for many environments configurations similar to above is the only way to deploy industry-accepted intrusion detection solution.

*Vladislav V. Myasnyankin, security expert, author and translator of several IT security related papers. At present time he works as Chief Information Security Officer at the bank. The author would like to thank Paul Pokrovsky and Michael Plotnikov for help with article development.*

*Anton Chuvakin, Ph.D., GCIA is a Senior Security Analyst with a major information security company. His areas of infosec expertise include intrusion detection, UNIX security, honeypots, etc. In his spare time he maintains his [security portal](#).*

## Relevant Links

[Complete Snort-Based IDS Architecture, Part One](#)

*[Anton Chuvakin and Vladislav V. Myasnyankin](#)*

[Privacy Statement](#)

Copyright 2006, SecurityFocus