Securing your network with snort intrusion detection

Brown, S.

Publisher's version / Version de l'éditeur:
https://doi.org/10.4224/8895478
Student Report (National Research Council of Canada. Institute for Ocean Technology); no. SR-2005-07, 2005

DOCUMENTATION PAGE

REPORT NUMBER	NRC REPORT NUMBER	DATE
SR-2005-07		April 2005

REPORT SECURITY CLASSIFICATION	DISTRIBUTION
Unclassified	Unlimited

TITLE
Securing Your Network With Snort Intrusion Detection

AUTHOR(S)
Stephen Brown

CORPORATE AUTHOR(S)/PERFORMING AGENCY(S)
Institute for Ocean Technology

PUBLICATION

SPONSORING AGENCY(S)

IOT PROJECT NUMBER	NRC FILE NUMBER

KEY WORDS	PAGES	FIGS.	TABLES
Snort, Intrusion Detection, Network Security	59	32	

SUMMARY

Networking is one of the most important aspects in modern computing. It allows us to quickly and effectively share ideas and work together on a common project. But as computer networks grow beyond simple home networks into the large corporate, national, and international networks of today, you can't trust everyone your computer is connected to. Enter Intrusion Detection: an Intrusion Detection System allows us to identify malicious network activity as soon as it happens, providing an effective means for preventing attacks from being successful. Snort is one of the leading standards in Intrusion Detection. It is an open-source project with a vast community of supporters constantly updating and improving the already exceptional piece of software. If you want to know how your network's resources are being used, Snort makes it possible.

ADDRESS National Research Council
Institute for Ocean Technology
Arctic Avenue, P. O. Box 12093
St. John's, NL A1B 3T5
Tel.: (709) 772-5185, Fax: (709) 772-2462

National Research Council Canada
Institute for Ocean Technology

Conseil national de recherches Canada
Institut des technologies océaniques

SECURING YOUR NETWORK WITH SNORT INTRUSION DETECTION

SR-2005-07

Stephen Brown

April 2005

Table of Contents

List of Figures

Securing Your Network With Snort Intrusion Detection

1 - Introduction

1.1 - What is Intrusion Detection?

An Intrusion Detection System (IDS) is responsible for identifying anomalous or inappropriate use of computer and network resources. Since intrusion detection takes on such a broad definition, there are many different approaches to achieve these results. The most common methods of intrusion detection involve statistical analysis of network activity and pattern matching. The method of statistical analysis involves monitoring network traffic over time and establishing a baseline for what types of traffic are normal. Abnormal traffic is then flagged as potential attacks. An IDS that employs pattern matching searches all network traffic that it sees for known attack patterns. For example, a common port scanning technique is to send FIN packets to all ports on the destination. A FIN packet is a TCP packet that marks the end of communication on the specified port and is part of the communication protocol. However, due to the way networking is developed in many operating systems, closed ports will respond to a stray FIN packet with a RST (reset) packet, while an open port will not respond at all. A pattern based IDS can detect stray FIN packets and alert the administrator of a potential port scan.

1.2 - What is Snort?

At its core, Snort is a rule-based Intrusion Detection System. However, due to the vast flexibility placed in Snort's design, it is capable of much more. According to Snort's website, it is "an open source network intrusion prevention system, capable of performing real-time traffic analysis and packet logging on IP networks." The flexibility offered by Snort comes in three main forms: the rule-based design, the option to use plug-ins, and the potential for add-on utilities.

A large part of Snort's Intrusion Detection functionality stems from the rather large library of rules that have been written for it. Since Snort uses pattern-based Intrusion Detection, it needs some way of knowing what patterns to search for within the network traffic it can access. Snort's rules provide this data. The language for detection rules is easy to understand yet powerful enough to detect many types of attacks or unacceptable network communication. And since you are given the option to write rules that detect the type of traffic you are interested in, you can always extend or alter Snort's default operation to log any type of traffic that may be of use in certain situations.

Snort also supports the notion of preprocessors. Preprocessors are tricky to define since they can perform many different tasks. They are applied to packets after they have been decoded and before they are handed to the detection engine where pattern-based matches are tested according to the rule set. Some common preprocessors include the ability to reassemble data that has been fragmented across multiple packets, and to detect intrusion attempts by means of sending packets that do not conform to standard protocol. Neither of these functions are possible

to achieve through rules, since the detection engine is simply a pattern matcher. Preprocessors provide another layer of flexibility to Snort as a whole since they provide an opportunity to perform more complex operations and tests. They can even signal alerts, and therefore can be used to detect attack patterns that do not leave standard signatures in data packets. This eliminates a flood of false positives that would be created by writing broad detection rules in attempt to capture such non-standard attacks.

Finally, Snort also gains flexibility from its compliance with add-on utilities. Snort allows alert logging to many standard formats including plain text and SQL databases. It has support for the most common SQL database systems including MySQL, PostgreSQL, and, on Windows, MSSQL. Such convenient output options facilitate the creation of analysis utilities that read Snort data and make it easier for an administrator to effectively deal with potential security threats. Some such utilities include ACID (Analysis Console for Intrusion Detection), SnortSnarf, PigSentry, among many others.

Thanks to the large community that use, support, and develop Snort, it has become much more than the “lightweight” IDS that the author described the first release to be. Snort is a major contender on the IDS front, and holds up quite well when compared with practically any free or commercial IDS alternative.

2 – How it Works

2.1 – Software Architecture

Unlike many projects of this scope, Snort does not operate on the client/server model. Because of this, a user needs nothing more than a connection to a network system with Snort on it to be able to start sniffing packets. For this, and reasons that will become apparent shortly, it is essential that any system with an installation of Snort be sufficiently secured.

Snort detects intrusions by listening to as much network traffic as it can to detect aberrant patterns. Upon installing Snort, it must be bound to one or more network interfaces. Snort will use all traffic on the selected interface(s) as the basis of its intrusion detection. For the purposes of intrusion detection, the interface(s) can operate in one of two modes: normal or promiscuous. The normal behavior of a network interface is to discard any packets in which either the MAC or IP address does not match that of the interface. In terms of the OSI reference model, the packet would be received by layer 1, and passed to layer 2 where it would be promptly discarded if the physical address was incorrect. This standard procedure is circumvented if the NIC is set to operate in promiscuous mode. In this mode, any packets seen by the interface will be passed up through the layers of software regardless of any addresses attached to the packet.

Once Snort receives a packet, it is sent through several stages of processing in order to determine whether or not the communication is acceptable by the standards set out in your configuration. First the packet is sent through the appropriate decoder. Decoders do the initial basic operations on network traffic. They decode the link layer data and gather information on the packets structure to pass to the next layer. This ability is based on the libpcap library and supports a number of link layer technologies including Ethernet, 802.11, Token Ring, FDDI, Cisco HDLC, SLIP, PP, and PF by OpenBSD. Decoders do not handle higher layer protocols, only link layer.

When the appropriate decoders have been run and have assembled data on the packet, it is then passed to the preprocessors. The preprocessors perform a number of different functions that have a rather wide range. A list of the more pertinent preprocessors and their basic function can be found in appendix A. If you take a quick look at the list you should see that preprocessors are capable of many different things. They can be used to decode or prepare data for the detection engine, reassemble fragmented data transfers, track long conversations using states, or even detect non-standard attack patterns that would be impossible to detect accurately using the rule language alone.

After a packet has been passed through the preprocessors, it at last reaches the detection engine. The detection engine is the core of what Snort has always offered from it's very beginning. The detection engine uses the rules that the administrator has assigned for network traffic to decide if what it is seeing is legitimate network usage. Snort is currently on the second major version of the detection engine. The first used a three-dimensional linked list to sort rules based on rule headers, rule bodies, and detection functions. As the number of standard rules increased, this method became too slow to cope with increasing network traffic and the event of gigabit over Ethernet. In the old detection engine, rule sorting was not sufficient to allow most packets to avoid completely irrelevant security checks. The new detection engine corrects this by sorting rules in four main groups based on types of traffic: TCP, UDP, ICMP, and IP. With a

much more focused search path, network traffic is not needlessly passed by all available rules. It only reaches the ones for which there is potential to trigger an alert.

The detection engine can generate output of several types, each of which can have separate destinations. The most common type of output is an alert. Alerts are security warnings of potential attempts to compromise a network. For administrative purposes, there are also two more options for keeping track of data: logs and tags. Logs are set up similarly to alerts in that they use the same rule language, except that the keyword *log* is used rather than *alert*. Logging allows you to capture specific data specified by a custom rule, but at the same time, keep that data separate from alerts. Tagging is a good option if you wish to perform statistical analysis on normal network traffic. Tag rules are evaluated in the post-detection phase are only triggered in the event the given packet did not cause an alert. If an attacker attempts to compromise a network, the initial attack should be caught by Snort. But the traffic they generate after a successful compromise may appear to be legitimate traffic. Using tags can be an effective way to verify whether or not an attack was successful.

There are many options available to Snort users. Snort's design is very broad and expandable, and there are even more options available to those who want to perform more specific or less standard tasks. For example, Snort also permits custom detection plug-ins, preprocessors, and output plug-ins. However, since the average network administrator would not use such options, they are not important to this discussion. These options do emphasize Snort's open architectural design. This expandability is precisely what has made Snort into what it has become. The Snort community is dedicated to providing any and all necessary functionality to detect attacks as soon as they happen.

2.2 - Sniffing Packets

The effectiveness of Snort depends heavily on its placement within a network's architecture. In fact, placement matters so much that it can completely change the base functionality of Snort. With regards to the scope of the data dealt with by Snort, there are two main categories: the Snort installation will either be a network-based IDS or a host-based IDS. With regards to the response taken when an alert is triggered, Snort either becomes an Intrusion Detection System, or an Intrusion Prevention System. We will now look at the basic types of Snort configurations, and the placement and usefulness of each.

2.2.1 – Network-Based Intrusion Detection

A network-based Intrusion Detection System listens to all available traffic on the network for potential threats. This type of IDS has been the topic of our discussion so far since it is the most common solution for a business network type situation. In order to get a network-based intrusion detection system to work, there are a few issues that we first need to overcome.

First, since the default behavior of a network interface is to discard any data received that is not destined for that host, we need to override this standard by setting the interface to promiscuous mode. Thankfully, Snort takes care of this for us. Snort will set the interface(s) to which it is bound to promiscuous mode on startup, and set them back to normal mode if the program is terminated.

Second, we need to make sure that all network traffic is sent to the computer with Snort running. This issue is trickier since it depends on many factors, some of which include the

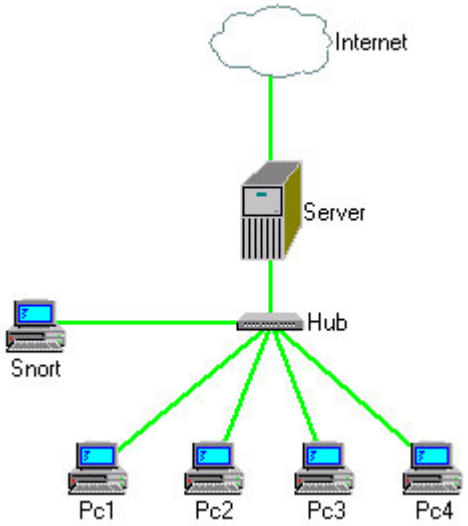

Figure 2-1: A simple network.

desired results of the IDS, the topology of the network that Snort is being placed in, the types of networking devices being used, which subnet(s) should or should not be listened, and how paranoid you want to be about discovering certain types of attacks. Let's take a look at a few examples to clarify some of the available options.

Figure 2-1 shows a simple network consisting of 4 workstations and a server connected together by a hub. Hubs are passive layer 1 devices and cannot make switching or routing decisions. They pass all traffic to all ports hoping that one of those ports is the destination, or at least knows where the destination is. An IDS can use this to its advantage. By connecting the IDS to the hub and setting the NIC to promiscuous mode, it will see all network traffic that has to pass through the hub.

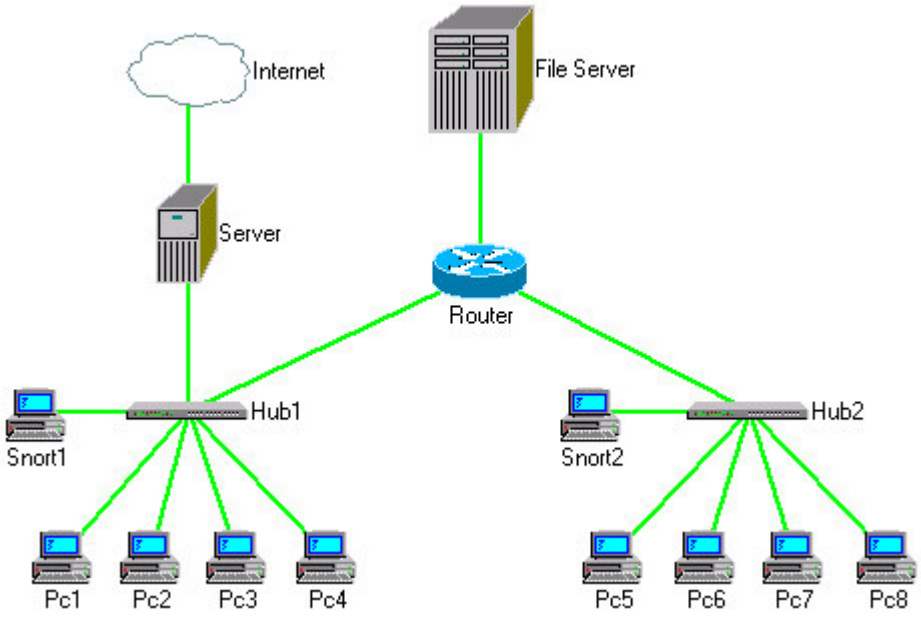

Figure 2-2: A network with a router.

Things get more complicated when layer 2 or layer 3 devices are used in the network. Layer 2 devices, such as switches or bridges, make decisions on where to send incoming packets based on the layer 2 physical address attached to the packet. Similarly layer 3 devices, routers, route packets based on layer 3 logical addresses. In either case, incoming messages are sent out only on the port that contains the message's destination. This poses a problem for network-based Intrusion Detection since our goal is to see all network traffic so that we can determine whether or not it is appropriate use.

Consider the network shown in figure 2-2. In this case there is a router segmenting the network into two main parts. The server that handles Internet access is located on the first network along with several workstations. The second network segment consists solely of workstations, and there is a common file server located directly on the router. If we place an IDS on Hub2, we will not see if any attacks are made against any computers on Hub1. If we place an IDS on Hub1, we will see all external traffic since Internet access is located here. However, we will not be able to tell if a computer on Hub2 attempts to compromise another computer on Hub2 or the file server. This may or may not be a concern, but if it is, a single IDS in either of these locations is not going to be sufficient.

In a network as small as the one shown in figure 2-2, maintaining two separate IDSs may not be a problem. However, placing an IDS on every network segment becomes an issue when the network expands to the complexity of that shown in figure 2-3. In this case, if you were to use the approach of placing an IDS on every network segment, you would have the task of maintaining 8 IDSs. Even if you trusted the people using your network and only wanted to monitor the segments on which there were important resources like the SQL server and the wireless access point, you would still have 3 IDSs and you wouldn't still wouldn't be keeping track of the file and web servers. Fortunately there are other options.

Many switches and routers allow the option to copy all traffic to a specific port, regardless of whether or not it is bound for that port. This copy is done without interrupting the normal flow of traffic: everyone else gets their messages, but you get them too. This ability is

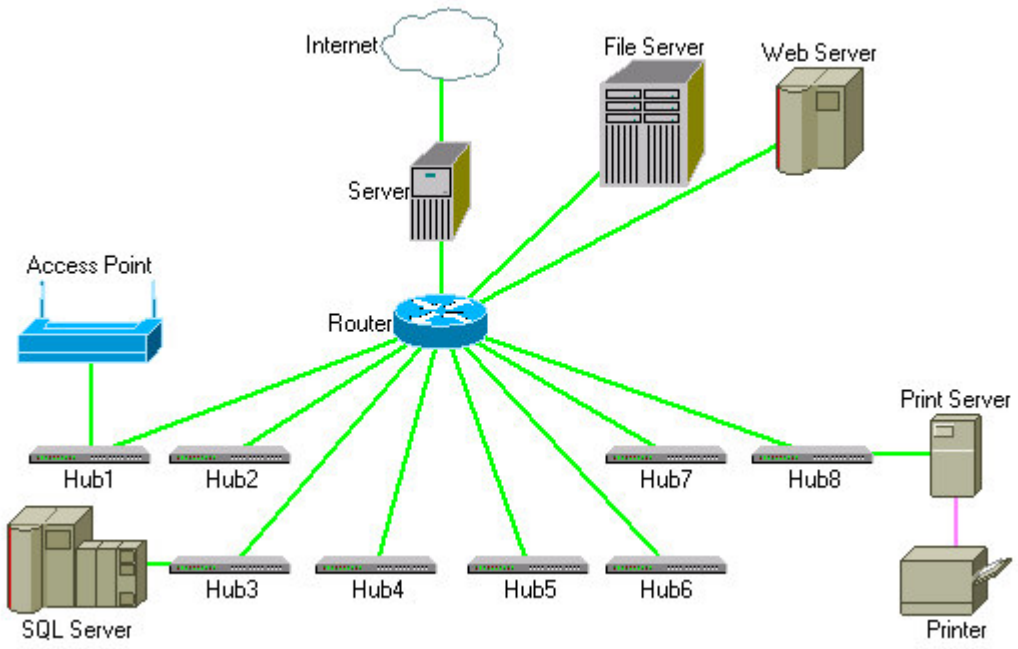

Figure 2-3: A complex network.

known as port mirroring or port spanning, as Cisco calls it, and is present on most large switches or routers. So, in the case of figure 2-3, we could place a single IDS on a spanning port on the router and it would see all network traffic (internal and external). It would even see traffic to and from the file and web servers, which we were previously unable to access.

2.2.2 – Host-Based Intrusion Detection

In addition to network-based Intrusion Detection, Snort also supports the notion of host-based Intrusion Detection. A host-based IDS detects intrusions on a single host only, rather than on a network. The idea of host-based Intrusion Detection may seem more natural in a home PC setting, but it also has its uses in larger networks. For example, if a specific computer is suspected of being compromised, a host-based IDS can be installed and may provide a clearer look at what is going on inside the box than a network-based IDS could.

Host-based IDSs can also serve statistical purposes. Snort can be installed on many hosts across a large network and set to run a custom rule set. They can all log data to a common destination, and since the rule set is customized, it would require updating only when the administrator's statistical needs change.

The main configuration difference between a NIDS and a HIDS is that a NIDS listening interface must be set to promiscuous mode, and a HIDS listening interface should be kept in normal mode since network traffic not pertaining to the local computer is irrelevant in this situation.

2.2.3 – Intrusion Detection vs. Intrusion Prevention

Originally an Intrusion Detection System, Snort's abilities have grown extensively and it is quite capable of operating as an Intrusion Prevention System (IPS). The names give a pretty clear indication of what each system does: an IDS will listen quietly and let you know if it thinks there is an intrusion, while an IPS will take an active response to actually stop the traffic from being transmitted. In order for an IPS to operate, it must be placed in a much more central place

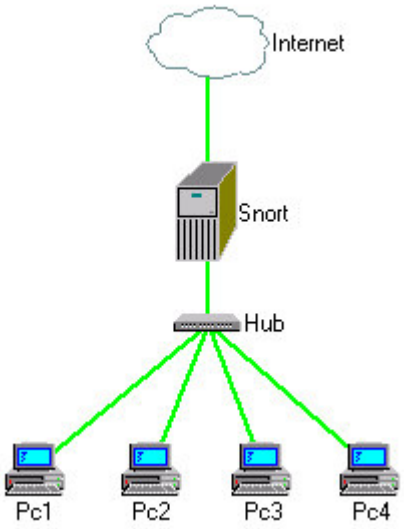

Figure 2-4: Snort as an Intrusion Prevention System.

on the network. An IPS must have potentially threatening traffic pass *through* it, rather than have traffic copied to it. That way, any traffic that Snort deems threatening, it will simply drop.

For this reason, IPSs are much more difficult to maintain. A standard rule set can generate hundreds of false positives a day on a busy network. If Snort were installed as an IPS without any work done on the rule set, a significant amount of traffic would be dropped, whether it was malicious or not. When setting up an IPS, it is necessary to spend a great amount of time fine-tuning the rule set so that normal and acceptable daily traffic is unaffected by the presence of the IPS, but attacks are still caught and intercepted with a high degree of accuracy.

Placing Snort as shown in figure 2-4 does not automatically make it an IPS. Snort can still run as an IDS if it is placed inline with network traffic. In fact, on a large multi-segment network, it may be one of the most efficient ways to place your IDS. To enable Intrusion Prevention, snort_inline has to be set in the configuration file, and iptables has to be installed on the system. By enabling snort_inline, three new rule types become available in addition to the standard alert and log types. The types drop, reject, and sdrop can be used to write rules for Intrusion Prevention.

2.3 – Logging Data

After Snort has triggered an alert or log event, the data goes through two more stages before being recorded: thresholding and suppression. These two stages allow you to limit the frequency of a given alert and to back out of logging at the last minute. They are basically for more fine-tuned results from your IDS and are in place to allow flexibility. After these two stages are complete, the alert is finally passed to the output plug-in.

Output plug-ins were introduced into Snort in version 1.6, prior to which alert output was hard-coded. With the event of output plug-ins, greater flexibility is provided to the system administrator on how data is recorded for viewing. You can chose from one of the many standard plug-ins that programmers have written, or even have the option of writing your own custom plug-in with the open output plug-in application programming interface (API).

This notion should already look familiar since we have seen that Snort allows many parts to be custom-programmable, like preprocessors. As Snort has developed, it has allowed for an increasing amount of choice by its users, allowing it to do many more things than a simple IDS could.

There are many output plug-ins available for Snort, including ones for various SQL databases. Logging security information to a database is a good idea since databases inherently require data to be structured, making data retrieval more organized, convenient, and specific to what the user requests.

2.4 – Securing Snort

Snort is all about making sure your computer resources are not being compromised. Placing your faith in your Snort computer to detect any break-in attempts, however, may not be a good idea if the Snort computer itself is not secure. You can be sure that in order to be a successful hacker, you need to know what the opposition is using to prevent you. And so, hackers have developed tools that will try to to either outsmart or suppress your Intrusion Detection System without it ever letting you know anything went wrong. Obviously, securing your security monitor is essential.

2.4.1 – Making Snort Invisible

One of the first things an attacker of a large network might do is listen a little before doing extensive probing to see what traps you may have set up on your network. The more information they have about your network structure, the more dangerous they can be. If you are running Snort to listen to all network traffic and logging alerts to a separate computer with a MySQL database, an attacker may probe some ports while listening to network traffic, see that some computer is sending security alerts and discover your Intrusion Detection System.

Of course, you may think this may be a little too paranoid, but the amount of security you need to apply to your network depends on your likelihood of being attacked. And if your computer or network is connected to the Internet, you can be sure that someone is going to be probing your security level at some point regardless of how useful you think your resources are to a hacker. In any case, since Snort generates sensitive information, some extra security doesn't hurt.

Firstly, it is not a good idea to allow the computer running Snort to send data on the same network interface as it is listening. For one thing, intermittent sending may cause you to lose some incoming packets on a busy network. More importantly, it can lead a hacker directly to your IDS and leaving open communication ports raises the risk of an attacker compromising the system running your IDS. There are a few solutions to this problem, depending on how critical you deem it that no signal is sent on this line. One effective solution if your IDS is connected to a router with a spanning port is to drop all outgoing messages from the IDS on the router. If this is not an option, there are other options available. You can try to manually lock down communication on the system side. This may work for most things at the Application layer, however it would be nearly impossible to stop the interface from inadvertently responding to a ping or maybe even a FIN scan at the lower layers of processing. You can also run your IDS on a network interface without a configured IP address. This will again reduce the adapters possibilities when responding to probes. However, if you really want to ensure that no one outside your network knows that this computer even exists, a one-way cable may be the best solution. For more information on what a one-way cable can do for you, as well as a quick guide on building such a cable, see appendix B.

Of course, a one-way cable is not required for Intrusion Detection. My purpose is here is simply to make you aware of the issues present when running an IDS. The data it deals with could tell a clever hacker about security holes present in your network so securing your Snort computer in some manner is a must. How much security you add depends on how much you think is reasonably necessary given your network size and nature.

2.4.2 – Viewing Snort Logs Securely

Due to the vast flexibility of Snort's design, many additional programs and plug-ins have been created that make dealing with alerts much easier. An untweaked installation can generate hundreds or even thousands of alerts a day on a busy network, so browsing them all in plain text is often not very convenient. There are many programs that will interact with the standard database used by either snort-mysql, snort-pgsql, or any of the other database output plug-ins, to display alerts in a more human-readable format. Acidlab, SGUIL, Snortplot, SnortSnarf, and PigSentry are some such programs, to name just a few of the many available. The setup guide in

the following section will cover how to install and configure Acidlab, a PHP driven web interface.

The most secure way to view Snort logs is by using the machine locally. However, this isn't always the most convenient or quick option, and sometimes isn't an option at all. Remote access is good if multiple administrators will be viewing logs, or if you need to quickly access logs from anywhere in order to deal with a security threat. But enabling remote access means ensuring that only approved people can gain access to this confidential information.

First, remote access to the computer should not be done on the network interface that is listening. As described in the previous section, the listening interface should only be listening. Any other communication can cause loss of packets, and poses the security risk of potential hackers discovering your counter-measures. If you want remote access to Snort logs, you should have a second network card in the Snort computer. Placing the second interface on a separate subnet or less prominent network location will further reduce the risk of discovery: the computer should look like any other network computer from this interface.

If you are logging data to a plain text file, or are using a database without any of the types of programs listed above, then you simply need to connect to the computer to browse alert information. Make sure to use SSH so that alert information is not transmitted in plain text across the network.

If you use Acidlab, or another such interface, then you can secure access by using SSL and password protection. The method for securing an Acidlab installation will be covered in the guide in the following section.

3 – Setup Guide

Snort is only dependent on a few standard libraries, all of which have been ported to multiple platforms, and so Snort too is available for many platforms. While the installation process may be different on various platforms, configuring the system is quite similar regardless of you choice of operating system. This guide will walk you through the installation of Snort and related software on Debian Linux. Installing Snort on Microsoft Windows is very straight forward, and configuring the system is common across all platforms.

3.1 – Before You Begin

3.1.1 – What is Apt?

Apt, short for “Advanced Package Tool”, is the very convenient update system built by Debian. It can perform software upgrades, installations, and management with minimal user supervision. One of apt's greatest features is its ability to automatically detect and resolve software dependency issues. In general many Linux packages have a great number of interdependencies, and without a package manager such as apt, these complicated dependencies must be sorted and resolved manually.

Apt also groups software based on the level of testing and confidence placed in each package. Linux packages, many being community projects, are updated very frequently, though several versions are kept since older versions may be known to work for sure, while newer versions may have added features that haven't been fully tested yet. Apt takes this concept and applies it by specifying three categories of software: stable, testing, and unstable. Stable software is known to work with all other stable software without question. Unstable is bleeding edge technology, and certainly not guaranteed to work: in fact it is quite likely to break if you use the newest features. Testing is the middle ground between the two. It is much newer than stable, but is less likely to break than unstable. The default behavior for apt is to update and install software that is stable, however this can be overridden with a method called pinning which we will look at shortly.

At the time of writing, the most up-to-date “stable” version of Snort is 1.8.4beta1-3.1. The current testing version is 2.3.0-7 and unstable has reached version 2.3.2-1. While installing any version of Snort beyond the stable version will require that you move some more of your packages beyond stable as well, there is good reason for choosing the testing version as your IDS. There have been many significant changes in Snort between versions 1.8 and 2.3, most notably the rewrite of the detection engine which is now much faster and can handle higher traffic networks. There is also the standard rule set to consider, which is much more comprehensive at detecting standard attack patterns with every new release. For a standard IDS it is definitely recommended to go with as new a version of Snort as possible.

3.1.2 – Getting Ready to Install

In order for apt to allow you to install anything that is not considered stable, you need to set up pinning. When installing or updating software with apt, it will calculate the benefits of moving to newer software based on a set of weights that can be manually edited. Each of the

three software categories (stable, testing, and unstable) can be assigned a weight, and the ratio among these weights will determine how necessary you think it is for apt to install software from either category. To alter pinning settings manually create the file `/etc/apt/preferences`. The following is a sample preferences file:

```
Package: *
Pin: release a=stable
Pin-Priority: 700

Package: *
Pin: release a=testing
Pin-Priority: 650

Package: *
Pin: release a=unstable
Pin-Priority: 600
```

When you invoke apt to do either an update or an install, the first thing it does is to query known Debian sources for the requested packages. There are many sources for Debian packages, and new sources appear frequently to support various packages. Debian.org is the most central source for packages, but it is not the only one, and may not even be your preferred one if you discover a faster mirror. Apt allows you to specify a list of potential sources for packages with the file `/etc/apt/sources.list`. Unless you have already modified this list, you should see entries like the following in this file:

```
#APT SOURCES LIST
deb ftp://ftp.us.debian.org/debian/ stable main non-free contrib
deb-src ftp://ftp.us.debian.org/debian/ stable main non-free
contrib
```

These entries identify debian.org as a source for stable packages. This guide will focus on a testing installation of Snort so you will need to add the following entries for apt to locate testing packages:

```
# TESTING SOURCES
deb http://http.us.debian.org/debian testing main contrib non-
free
deb http://non-us.debian.org/debian-non-US testing/non-US main
non-free
```

These entries can be modified to point to your favorite mirror that contains testing packages.

There is one more step that may be required before you can install Snort. If you take a look at the contents of `/etc/apt/apt.conf` you should see a line like the following:

```
APT::Cache-Limit "8388608";
```

This specifies the maximum amount of memory that apt should use while it is running. If this memory cap is exceeded, the program will terminate. Since we have added more sources to

our list, apt will require more memory to operate. Apt uses this memory to get lists of packages at each source, sort them, and create dependency trees for each of the requested packages. This can be a fairly large operation, particularly if you have a large number of sources in addition to the four listed above or are planning on upgrading a number of packages to testing status. If apt does not successfully complete the installation you may need to increase the memory available to it. The following allowed me to complete the installation:

```
//APT::Cache-Limit "8388608";
APT::Cache-Limit "10000000";
```

Note that the default was commented rather than deleted or replaced so that it could be returned to it if the new option ever caused any problems.

3.1.3 – Snort Components

Before we install Snort, here's a quick note on the components that make it up. As said before, Snort offers several options for data logging. It can log to plain text or many standard types of databases including MySQL and PostgreSQL. You need to know how you plan to do your data logging when you install Snort as it will dictate which packages you will get. The following is a list of Snort packages available to Debian through apt and a short description of their function:

snort:	This is the Snort IDS, but this package only supports output in plain text.
snort-common:	This package is necessary for all Snort installations, regardless of output format.
snort-doc:	The Snort documentation.
snort-mysql:	Snort, with the added ability to output to MySQL.
snort-pgsql:	Snort, with the added ability to output to PostgreSQL.
snort-rules:	A virtual package that points to snort-rules-default.
snort-rules-default:	The default rule set.

The three packages snort, snort-mysql, and snort-pgsql are mutually exclusive of one another: you can only have one of them installed. Snort-common is required, regardless of which of the three you select.

In this guide we will set up a Snort IDS that will log to a MySQL database.

3.1.4 – Other Packages

In addition to the Snort components we will be installing several other pieces of software that will help us with intrusion detection. Here is a list of these programs with a short description of their function:

MySQL:	An SQL database system. We will use MySQL for logging Snort alerts in a central, secure location.
Acidlab:	ACID stands for Analysis Console for Intrusion Detection. It is a PHP script that queries the standard MySQL database that Snort uses for logging and performs

various sorting, formatting, editing, and analyzing functions. In short, it makes dealing with Snort alerts much easier.

Apache-SSL: Apache is a very widely accepted web server. Since Acidlab is a PHP website, we will use Apache to run a web server that will host Acidlab. SSL stands for Secure Sockets Layer, and provides a secure encrypted connection between two computers. Since the information Acidlab passes out is deals with security vulnerabilities, it is important that it is not seen by the wrong people. We will use SSL to secure all communication with the Apache web server.

PHP: PHP is a recursive acronym that stands for "PHP: Hypertext Processor". Since Acidlab is a PHP script, we will need the PHP processor to run Acidlab.

3.2 – Installing Snort and MySQL

Debian's Advanced Package Tool makes installing and configuring Snort and MySQL a snap. Handy configuration utilities alleviate the need to search through lengthy script files in search of options that must be set in order for the software to work. We will now look at a brief overview of the setup process, quickly highlighting some of the major "gotchas" in the system. For a complete step-by-step guide to the installation procedure, see appendix C.

First, it is easier if you begin by installing MySQL. This way, you can set your root password, create a database and a user account for Snort to use. The Snort configuration utility will ask you for this information, and if you don't already have it, you will need to reconfigure Snort later.

When you create a MySQL account for Snort, remember you should only give it the permissions it will require to operate. Giving more just opens a window of opportunity for hackers. The required permissions are `insert` and `select` on the entire database, and `insert`, `select`, and `update` on the sensor table which you will create later.

When you install Snort, the only step that may cause some confusion is the final one where you are told to build the database structure. Since the configuration utility runs before files are actually installed, the file referenced in the note will not currently exist. You have two options: complete the installation, follow the instructions, and restart snort; or, find the Debian package, extract the files to a temporary location, follow the instructions and complete the setup.

If you require more detailed information anywhere along the way, check appendix C.

3.3 – Installing a Second Network Card

I'm not going to go over all the details of installing a second network card, because the process completely depends on the version of Debian you are running and the type of network card you wish to install. There is one important note that should be mentioned here, though. You want to ensure that one network card will be constantly listening, and the second network card will do any sending that may be required. To do this, one necessary step is to isolate the listening interface on it's own subnet so the operating system will by default try to use the second device for transmitting messages. This is achieved in the `/etc/network/interfaces` files. An example of this file is shown below:

```
# /etc/network/interfaces -- configuration file for ifup(8),
ifdown(8)
```

```
# The loopback interface
auto lo
iface lo inet loopback

# The first network card - this entry was created during the
Debian installation
auto eth0
iface eth0 inet static
        address 10.1.1.1
        netmask 255.255.255.255
        network 10.0.0.0
        broadcast 10.1.1.1
        gateway 10.1.1.254

auto eth1
iface eth1 inet static
        address 10.1.1.2
        netmask 255.0.0.0
        network 10.0.0.0
        broadcast 10.255.255.255
        gateway 10.1.1.254
```

As you can see, the `netmask` of `eth0` is `255.255.255.255`, meaning it is it's own network. In addition, the interfaces broadcast address is also set to itself. The second interface is set up in a standard way so that it will allow normal network communication.

3.4 – Installing Acidlab and Apache-SSL

Setting up Acidlab on an Apache web server is a little bit more involved than installing Snort, but if your are doing the job on Debian, Apt will most often make sure you know what needs to be done manually. We will now take a quick look at some of the less obvious things that need to be done while installing Acidlab. For a complete step-by-step guide to installing Acidlab, see appendix D.

Firstly, You're going to want to install Apache-SSL first. That way, when you install PHP and Acidlab, the configuration utilities will do most of the work for you.

There is one step involving setting up PHP that is neither done automatically, nor are you told it must be done until you see an ugly error message the first time you load Acidlab. In the PHP initialization file, the line that includes the MySQL shared object extension is commented by default. You will need to uncomment this manually.

Before you install Acidlab, you should create a MySQL account for it to use. Acidlab requires more privileges to operate than Snort does, since you can use it to manage the database, not just view it. The essential privileges are `create`, `select`, `insert`, `update`, and `delete`. Again, granting more than this is both unnecessary and not recommended. Acidlab also supports the notion of an alert archive which is a repository for alerts that have already been addressed but are still important to keep for reference. Details about this are given to you during the configuration. An archive database requires the same structure as the alert database, so simply create a second database, and follow the same procedure as before to build its structure.

Note that you will have to grant Acidlab privileges on the archive database as well if you choose to use it.

If you require further details on installing any of these programs, see appendix D.

4 – Running an IDS

Now that you know what Intrusion Detection Systems are all about and you've got one installed and configured on your network, you may be wondering “So what do I do with it?” We are now going to explore general maintenance issues, and talk about how to get the most out of your IDS. This section should provide an overview for tuning Snort's operation. It is accompanied by several appendices containing reference material. If you will be maintaining an IDS I recommend briefly skimming these references to get a feel for what options are available should the need for them arise.

4.1 – Start Sniffing

By default, Snort takes the initiative to grant itself the privilege of starting when you turn on your computer. This standard behavior can be circumvented by running the manual and more detailed configuration utility with the command:

```
dpkg-reconfigure snort-mysql
```

For a computer whose sole purpose is network-based Intrusion Detection, allowing Snort to load on startup is not a bad thing. But in either case it is still important to know how to load it manually. When Snort is running, you can not interact with it at all. You must tell Snort everything it needs to know before it is started. This is achieved through configuration files and command-line switches. We will take a closer look at the options available through the configuration file a little later, but the point is that the configuration files are read at start time, so changes are not picked up by snort unless it is restarted.

If you try to run Snort by simply typing `snort` at the command prompt, you will be shown a complete list of command-line switches along with the concise error message: “`Uh, you need to tell me to do something...`” The reason Snort is complaining is because you need to tell it the location of a configuration file to use. To do this, either `cd` to `/etc/snort/` and then run `snort`, or type the following:

```
snort -D -c /etc/snort/snort.conf
```

4.2 – Command-Line Switches

In general, Snort's command-line switches are used to change the way it behaves without changing the configuration file or creating a new one. Most all of the options available through switches are also available through the configuration file.

Two of the most important switches `-D` and `-c`. `-D` tells Snort to run in Daemon mode so you don't have to leave a terminal active displaying Snort data. `-c` allows you to tell Snort the location of the configuration file. You can avoid using this switch by running Snort from the directory containing the `snort.conf` file. Most of the other switches are only used for specialized runs of Snort, not for average daily use. To get a quick reference of the available options, you can run Snort with the `-?` switch, or see appendix E.

4.3 – Defining Behavior

Running an Intrusion Detection system is about striking a balance between keeping your set of rules broad enough to detect any attacks, yet precise enough so that you are not flooded with false-positives. In order to effectively maintain an IDS in this manner, it is important to know what goes into defining it's behavior. Snort's code of conduct is defined by two key influences: the configuration file, and the rules. We will now look at the structure of both.

4.3.1 – Snort.conf

The Snort configuration file is very well documented with useful comments interspersed throughout. Because of the thorough documentation, the file is rather long, so I will not reproduce it here. We will be looking at the general layout of the file to give you an overview of the options presented to you with a brief description of each section. If you would like to follow along with the actual configuration file, you should open it up on your computer.

The first section of the configuration file deals with network variables. These variables allow you to identify the various parts of your network. For example, the first two variables you will encounter are `HOME_NET` and `EXTERNAL_NET`. The default configuration file uses the most general values so a default installation of Snort will function on any system. However, providing Snort with more detailed information about your network allows it to make more informed decisions about the traffic it is seeing on your network. For example, a rule that decides whether or not a web page request is legitimate will probably use the `HTTP_SERVERS` variable to see if the destination address is in fact a web server.

Next, you will find a section dedicated to the Snort decoder. By default this section is completely commented and inactive, since the options here are for disabling pieces of the decoder. You can disable some of these options if you are finding that the decoder is generating too many alerts for valid network traffic.

The next major section deals with Snort's preprocessors. As discussed previously, preprocessors can do many different things. For a short description of the function of most of the common preprocessors, see appendix A. In general, the default configuration of each of the preprocessors is fine for intrusion detection. If, however, you need to either modify or fine-tune the performance of an individual preprocessor, you will need to seek more specific information on its usage. In the default configuration file, there is a very useful description accompanying each preprocessor, and in most cases a reference to a `README` file where you can find more details.

The next section is for configuring the output plugins. If you are running Snort on Debian Linux, the configuration utility that runs at setup time will complete this section of the configuration file, so you should only need to dive into this section if you need to change your output method or if you want to start logging packets.

The final important section is a list of rule files that Snort will use to detect attacks. If you look at the list, you will see there are several commented entries. All rule files that deal with detecting attacks are uncommented by default. However, included with the default rule set, there are also several rules that deal with company policy. If you wish to detect network activity that is against policy, these rule sets may be used.

4.3.2 – The Default Rules

There are currently 48 files included in the standard rule set, each containing one or more rules, accounting for a few thousand standard rules. These rules detect a wide range of attacks including port scans, buffer overflows, denial-of-service attacks, and software exploits. There are also several files containing rules that detect traffic that may contradict company policy. For example, if you would like to enforce a policy that no one may use chat rooms or instant messaging programs within your network, you can use the `chat.rules` file to detect such traffic. By default, on a Linux computer, the standard rule files are installed to `/etc/snort/rules/`. Take a look around the directory to get a feel for the types of rules present, or check appendix F for a complete listing of alert categories and the purpose. You can drastically modify Snort's behavior simply by enabling or disabling (via commenting) any of these rules.

As you can see, many of the rules that relate to a certain type of attack are written to use the standard variables defined in the configuration file. By specifying values for these variables, you can not only detect specialized servers running in locations they shouldn't, but also reduce the workload of Snort. If a rule is known to be only relevant to certain computers, Snort will not waste its time checking other computers against the attack signature. So, setting these values is a good way to boost the performance and effectiveness of your IDS.

4.3.3 – How to Read Snort Rules

As a Snort administrator, it is important to know the structure of the rule language. For one thing, alert log messages are far from detailed, so at some point you may need to read the rule to understand exactly what triggered the alert. In the event an alert is triggering to frequently, you may want to modify an alert rule to narrow its focus and reduce your alert count. Or, you may even want to write your own rules for statistical or security purposes.

The Snort rule language seems rather cryptic at first glance, but once you understand its structure it is very efficient and easy to read. The first requirement for a Snort rule is that it must be contained entirely on a single line: you can not place hard returns anywhere in the middle of a rule. Now let's break down a sample rule:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 113 (msg:"SCAN ident
version request"; flow:to_server,established; content:"VERSION|
0A|"; depth:16; reference:arachnids ,303; classtype:attempted-
recon; sid:616; rev:4;)
```

The above rule comes from the `scan.rules` file and detects a type of port scan. First, rules are divided into two main parts: the header and the options. The options are contained between parentheses; everything before that is the header.

The first things you'll notice in the header is the word `alert`. This identifies this rule as one that will generate alerts. If you are using Snort as an IDS then there are 5 options available to you here: `alert`, `log`, `pass`, `activate`, and `dynamic`. `Log` will send the packet to a log file rather than generating an alert. `Pass` will ignore the packet, and is useful when describing permissible traffic is easier than describing illegitimate traffic. `Dynamic` rules are ignored until they are activated by an `activate` rule. If you are running Snort as an IPS, you can also use the `drop`, `reject`, and `sdrop` actions.

The next piece of information is the protocol that the rule applies to. Snort currently supports four protocols: TCP, UDP, ICMP, and IP.

Next you specify one set of IP addresses. IP addresses must be identified in CIDR format. Multiple address ranges can be specified by separating them by commas, without spaces, and enclosing the entire sequence in square brackets. For example:

```
[10.1.1.0/24,10.2.0.0/16]
```

You can also use the negation operator ! to tell Snort you mean all other address besides the ones identified. You may also use Snort variables instead of IP addresses. For example, you can use `$HOME_NET` to mean the IP addresses present in the configuration file. Or, you can use the keyword `any` to mean any IP address.

Following IP addresses is the port number. You can tell Snort to check on a single port or a range of ports. To use a range of ports, separate the lower and upper bounds by a colon, like so:

```
50:100
```

You can also use the colon like this:

```
:100
```

To mean ports less than or equal to 100, or:

```
50:
```

To mean ports greater than or equal to 50. You can also use the keyword `any` to listen on all ports.

Next is the direction operator. This is either `->` or `<>`. This operator indicates the direction of flow for the packet. The first, `->`, means that the packet originates in the range to the left of the operator, and it's destination is in the range on the right. The other, `<>`, means that either could be the source or destination. Following the direction operator is another address and port range pair.

This now brings us to the rule options section. As of Snort version 2.3.2, there are 45 standard options. This number has dramatically increased over the course of Snort's revisions, allowing more flexibility in detection methods. The rule options section is made up of some combination of these 45 rules. Each option follows the same format:

```
option: data;
```

The format of `data` depends on the option being set. Options have been categorized into four main categories: meta-data, payload, non-payload, and post-detection. Meta-data options contain data that does not affect the operation of the rule, but is useful for determining the cause of an alert. Payload options are used for detecting attacks whose signatures can be identified within the payload of the packet. Non-payload options are also detection options, but they detect attacks by means other than searching a packet's payload.

The example given previously is an average pattern-matching rule. Many of Snort's standard rules follow this format, so we will break this example down to get a feel for how an average Snort rule is built. The first option, `msg`, is used to attach a user-readable message to an alert. This message is logged along with the alert when it is triggered. Acidlab will use this message data to identify an alert. This is considered a meta-data option because it does not affect the operation of the alert, but rather contains information useful for analysis. This message can contain anything, but by convention it usually contains the alert's category in capital letters at the beginning of the message (in this case the category is "SCAN"). The second option, `flow`, is a non-payload detection option that limits the rule to be applied to traffic that flows only in certain directions. Its function is similar to that of the direction operator found in the rule header, but it offers more fine-tuned control over the type of data flow to be analyzed. Next is `content`, which is a payload detection option. This option tells the detection engine a string to search for within the packets payload. Many attacks can be identified simply by a signature in a packet, so this option does the pattern matching. `Depth`, another payload option, tells Snort how deep to search for a `content` string. If you know an attack signature will always appear within the first 8 or 16 bytes of a packet, it is pointless to search any further since it will only be wasting processing power and will likely generate some false-positives. The remaining options are all meta-data and deal with identifying the attack and referencing its discovery documentation.

If you are interested in learning more about the Snort rule language, or need to check what a specific option does, the Snort manual has an excellent reference for the entire language. The current reference can be found at the web location: `http://www.snort.org/docs/snort_htmanuals/htmanual_232/node16.html`. Future manuals will have a different version number and name, so if a newer manual is available, you should find it on the main documentation page: `http://www.snort.org/docs/`.

4.4 – The Acidlab Interface

ACID stands for "Analysis Console for Intrusion Detection". Acidlab is a PHP script that interfaces with the standard database structure that Snort uses for logging alerts. It manages alerts, identifies relevant issues such as common alerts, and allows you to perform complex queries at the press of a button. We will now take a look at how Acidlab, or a similar analysis program, can make your life as the administrator of a NIDS much easier.

4.4.1 – Main Page

Acidlab, being a PHP script, is run through a web server. The way in which you access your Acidlab installation depends on how you configured your web server. If you configured Apache to be accessible only by the local host, then you will need to open up a web browser on the machine running Apache and type the address `https://localhost/acidlab`. If you are running Acidlab on an unsecure version of Apache, you can simply type `localhost/acidlab`. If Acidlab is accessible by other computers it is not recommended to be running Apache without SSL. See the setup guide for details on how to configure Apache-SSL. To access Acidlab on a remote computer with SSL, open a web browser and type the address `https://host.domain.extension/acidlab`, replacing `host` with the name

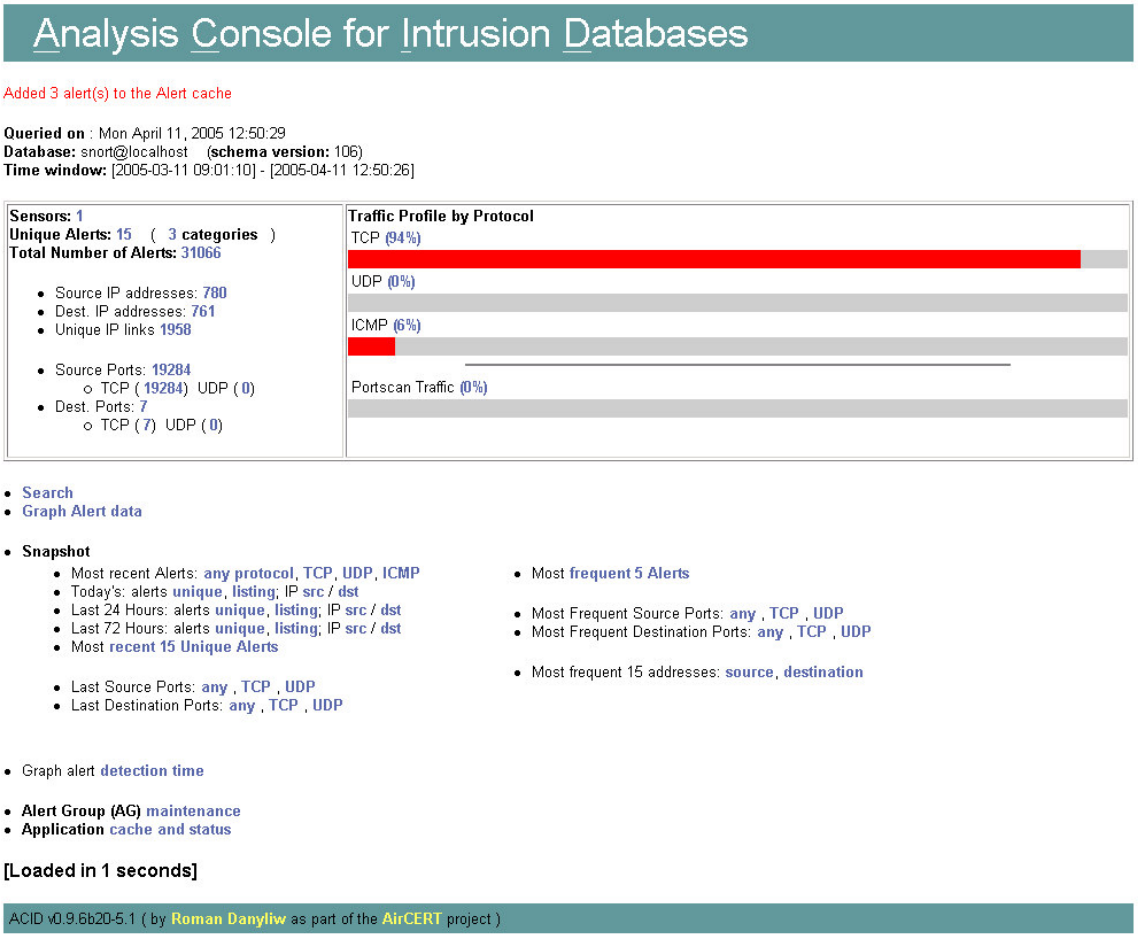

Figure 4-1: Acidlab main page.

of the `computer` hosting Acidlab, `domain` with your local domain name, and `extension` with the appropriate extension for your domain.

When you first open Acidlab, you should see something similar to figure 4-1. The main page shows you many quick statistics at a glance, and provides many opportunities for more in-depth analysis. The first thing you will see at the top of the page is a message written in red telling you how many alerts have been added to the database since your last visit. This is followed by three lines of information regarding the current date, time, and time window for which the new data applies.

Next is a table containing some useful quick statistics. First, Sensors refers to the number of listening interfaces found in the database. In our case, there is only a single sensor. This number will be greater than 1 if you either have multiple listening interfaces in your Snort computer, or you have multiple computers running Snort and logging data to a common database which Acidlab is using. Clicking on the number will bring you to a list of sensors found in the database, and give you the opportunity to do more detailed queries on each sensor's alerts.

Following sensors is the number of unique alerts in the database. In the event you either have a false positive left to remove, or you have been attacked with a DoS or anti-IDS attack, your database will be flooded with alerts. When browsing logs manually, this could cause you to miss an actual attack amongst the flood of other alerts. This provides you with a view of how

Figure 4-2: Acidlab title bar.

many different alerts have actually been entered in your database. You can either click this number, or the number of categories to narrow in on a search.

The next three statistics tell you the number of unique source and destination IP addresses that have appeared in any alert, as well as the number of unique IP links. Clicking any of this numbers will display a list of the unique addresses and allow you to search based these criteria. The final piece of information in this box is the number of unique source and destination ports appearing in the database. These links work in the same manner as those prior.

In the box to the right, you should see a bar graph break-down of all network traffic. This gives you an indication of which Snort-supported protocols are being used most frequently on your network. You can also use the links provided to search through alerts that originate only from a certain protocol.

Immediately below the information box are the search and graph links. The search option is extremely detailed and will allow you to search your alerts based on any criteria imaginable. We will look at it more thoroughly shortly. The graph page allows you to generate graphs based on several different criteria within any time frame you wish. This page requires PHPlot in order to function, which can be downloaded at www.phplot.com. If you installed Acidlab on Debian, apt should have taken care of the dependency issue automatically for you.

Following this, under the heading “Snapshot” you are provided with a series of links that allow you to view the most recent and most frequent alerts. Many of these links are very useful for daily intrusion analysis as they provide you with manageable size reports on the status of your network, with many opportunities to dive deeper and gain more thorough information on the nature of the alerts.

Near the bottom of the main page you will see a link to the alert group maintenance page. This page allows you to create, modify, or remove alert groups. Grouping alerts can be useful to keep data organized, particularly if you decide to start using Snort for statistical purposes. In this case you can keep statistical data separate from alerts that describe potential intrusions.

The final link on the main page provides you with some insight into the system driving Acidlab. It will provide you with the version of Apache and PHP you are using, as well as the status of your database.

Now, before we begin looking in more detail at some of the specific parts of Acidlab, there is one more noteworthy point. At the very top of the page you should see the title bar displaying the program's name. On any page other than the main one, this bar provides you with useful navigational links. An example can be seen in figure 4-2. The title bar provides you with the title of your current page as well as links to the home, search, and alert group maintenance pages.

4.4.2 - Search

As you can see in figure 4-3, the search page is loaded with options. You can search for alerts based on any type of information you want. The first group of options, “Meta Criteria”, allows you to perform a search on the alert meta data. You can search based on the sensor that detected the alert, the alert groups you have set up within Acidlab, the signature of the attack,

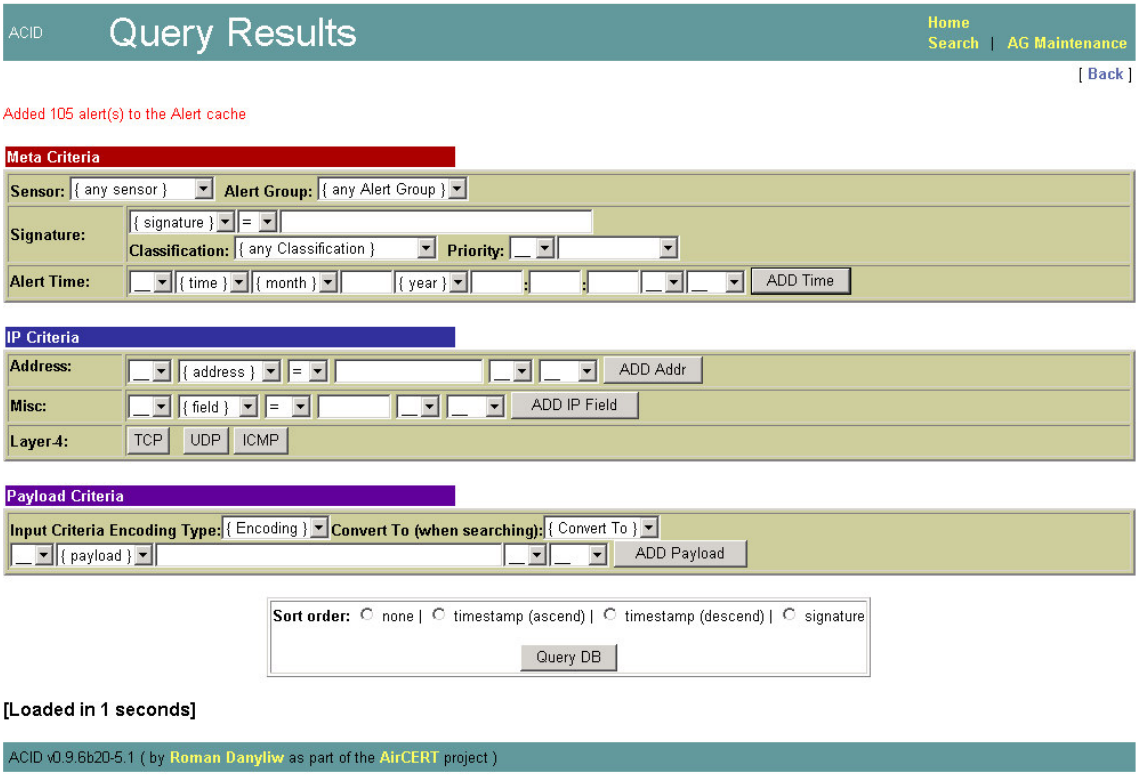

Figure 4-3: The search page.

and the date when the alert was signaled. You are even able to add multiple time ranges using the `ADD Time` button.

The second block, titled “IP criteria” lets you search based on addressing and protocol information. You can use controls specify one or more addresses, which may appear as the source, destination, or either in an alert. Clicking either of the three layer 4 buttons (TCP, UDP, and ICMP) opens up another block of options for more specific searches on these protocols. For example, if you pick TCP, you are able to specify ports for the search.

The final block let's you search based on the packets payload. You can specify one or more search patterns in either hexadecimal or ascii, and actively convert between the two encoding schemes rather than performing manual conversions.

It is evident the programmers of Acidlab took quite a lot of care in ensuring the search options provided maximum flexibility to the administrator. With this tool, you should be able to track down anything that appears in your alert database.

4.4.3 – Query Results

Acidlab provides very interactive results to any query. Rather than going back to the search page and refining your search pattern through the forms, you can dive deeper into your results on the fly through conveniently placed links. Figure 4-4 shows the results of a sample query. In this case, the most frequent 5 alerts are being displayed. As you can see from this query, alerts are identified by their signature. This signature is the data that was contained in the `msg` option in the rule. You should notice that most of the entries in the table are links. Clicking on any of these links will take you to a list of the things they describe. For example, the first alert in the list, whose signature is `NON-RFC HTTP DELIMITER` has `17089` instances of the alert

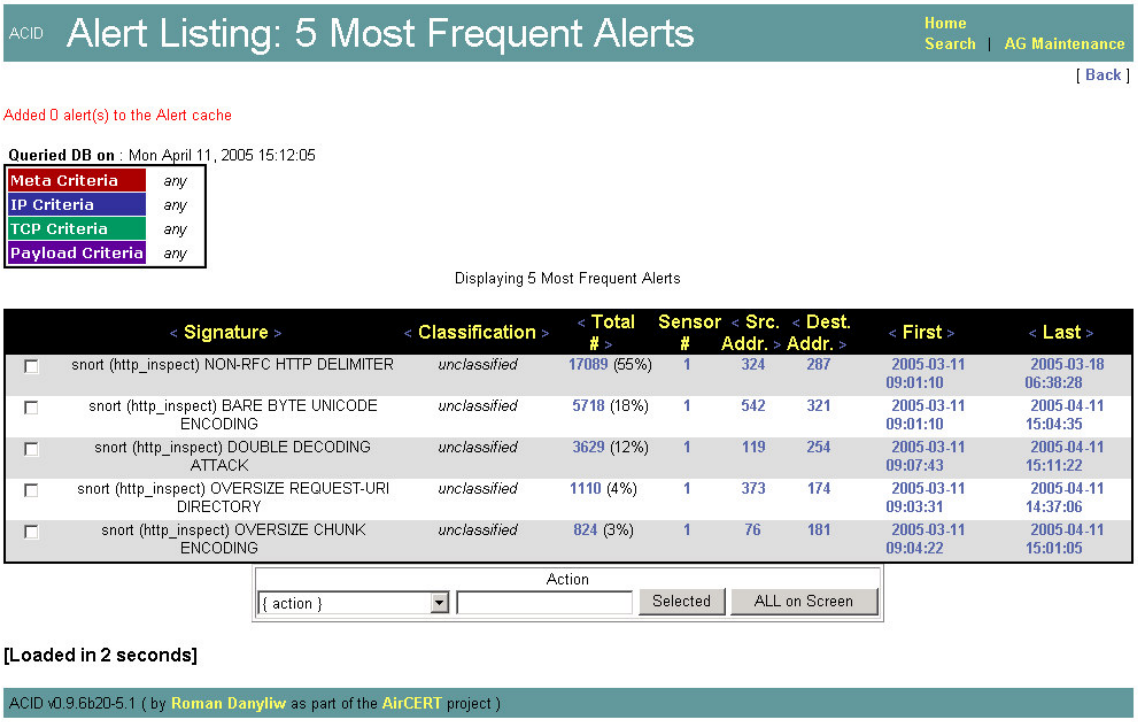

Figure 4-4: Query results.

in the database. Clicking on this number will take you to a query displaying each individual instance, complete with information on source and destination IP addresses. On that page, clicking on an IP address would take you to a query which displayed alerts involving that specific IP address. And so, you can navigate from one query to another to track down the information you require simply by using query results provided.

You can also get a detailed listing of a single alert, including the payload of the packet. Once you have narrowed your results down to individual alerts by some means, selecting any of those alerts will take you to the detailed listing. You will recognize a search whose results are individual alerts by the presence of a table column titled `ID`. Every alert logged in Snort, is given a unique ID number. In Acidlab, this ID number is a link to the alert's detailed information. There are, however, other ways to get to such a page. For example, in the query shown in figure 4-4, by clicking the date in either the column labeled `First` or the one labeled `Last` will take you to the detailed page for either the first or last alert of that type, respectively.

Acidlab's interface is very intuitive, and the programmers have paid very close attention to detail. You should never find yourself without a way to get the information you require when using Acidlab. For more information on Acidlab such as release notes, frequently asked questions, or community support you can check out Acidlab on SourceForge at `http://sourceforge.net/projects/acidlab`, or on the programmer's personal page at `http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html`.

5 - Conclusion

As technology progresses and more of our everyday activity takes place in the digital world, it becomes increasingly important to protect that world. Intrusion Detection has grown immensely in the very recent past, and it continues to grow thanks to the large and dedicated community of developers and security specialists that back it up. Rules that detect the latest and greatest hacker attack are often written within hours of their discovery, not days. As a network administrator, being able to take advantage of such an indispensable resource for free is incredible. If you ever find yourself stuck when it comes to network security, there are many users groups out there ready to help. In the fight against malicious network activity, we are not alone.

6 - References

1. Andrew R. Baker et al., eds., *Snort 2.1 Intrusion Detection*, 2nd ed. (Rockland, MA: Syngress Publishing Inc., 2004).

2. Apache Software Foundation, "Apache HTTP Server Version 1.3 Documentation", http://httpd.apache.org/docs/

3. Critical Networks, "Scalable Network Sniffing Architectures", http://www.criticalnets.com/resources/sniffscale.html

4. Fyodor, "Nmap Network Security Scanner Man Page", http://www.insecure.org/nmap/data/nmap_manpage.html

5. Laurie, Adam, Ben Laurie. "How to Create an Apache SSL Certificate", http://galwayland.com/notes/apachessl.htm

6. Pointon, Adam. "One-way Cable for IDS Deployment", Sentinel Data Security, http://sentinelsecurity.net/whitepapers/onewaycable.pdf

7. Sourcefire, "Inline Mode", http://www.snort.org/docs/snort_htmanuals/htmanual_232/node7.html

8. Sourcefire, "Writing Snort Rules: How to Write Snort Rules and Keep Your Sanity", http://www.snort.org/docs/snort_htmanuals/htmanual_232/node16.html

9. Stearns, William. "Building a one-way ethernet cable", http://www.stearns.org/doc/one-way-ethernet-cable.html

Appendix A - Preprocessors

The following is a list of some of the more pertinent preprocessors and their basic function:

frag2:	Defragments IP packets and launches alerts if a fragmentation attack is detected.
stream4:	Reassembles and inspects complete TCP streams. Through the use of states, this preprocessor can also detect some portscan and reconnaissance activity.
http_inspect:	Decodes HTTP communication by replacing hexadecimal character substitutions of the form '%00' to the ASCII characters that they represent. This facilitates further analysis of HTTP packets. Note that http_inspect was introduced in Snort 2.1 as a replacement for http_decode. http_decode will not work in Snort 2.1 and beyond.
rpc_decode:	Decodes RPC packets.
bo:	Identifies and alerts on any Back Orifice traffic.
telnet_decode:	Decodes telnet communication.
portscan:	Detects portscan activity. The default behavior is to alert if there is scan-like activity found on 4 or more ports in a window of 3 seconds or less.
portscan2:	Detects portscans by tracking TCP, UDP, and ICMP communications. The conversation preprocessor is required for this preprocessor to work.
conversation:	Required for portscan2 to function.
flow:	A state-based flow tracking preprocessor used by many of the standard rules that come with snort.

Appendix B – One-Way Cabling

A one-way cable allows a computer to receive data but not send any. Figure B-1 shows a diagram of the required pins in a standard 100 Mbps cat-5 cable. On the computer side, pins 1 and 2 are used for transmitting data, and pins 3 and 6 are for receiving. The most basic way of creating a one-way cable is to simply cut the first and second pins, the transmission lines, as shown in figure B-2. This may or may not work depending on your networking technology since some more sophisticated devices will check for an active signal rather than potentially wasting CPU cycles on transmitting a message that it thinks will not even be received. In these cases a better cable design is required.

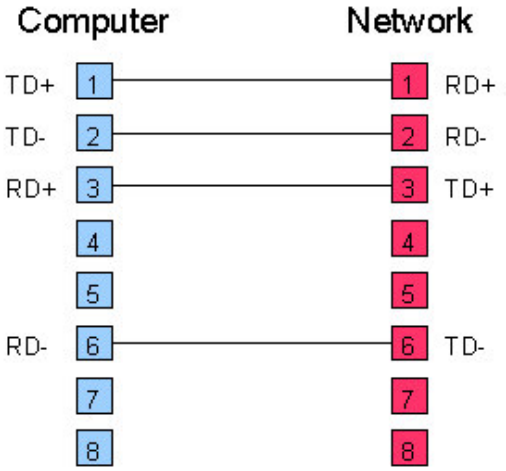

Figure B-1: A standard straight-through 100Mbps cat-5 cable.

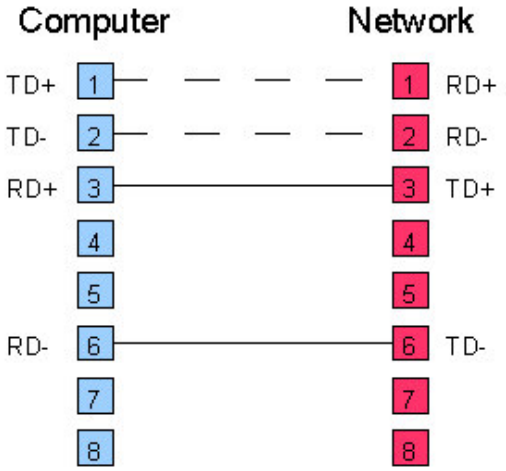

Figure B-2: A simple solution to making a one-way cable.

Another solution that works in some cases is shown in figure B-3. By reversing the two transmission lines, signals will still be sent and a connection maintained, but on some networking devices the signal would be unrecognizable due to the reversal of most data. Again, this will not work in all cases since some networking devices are smart enough to recognize the faulty wiring and correct the signal.

A third design is shown in figure B-4. Several people who have written guides to one-

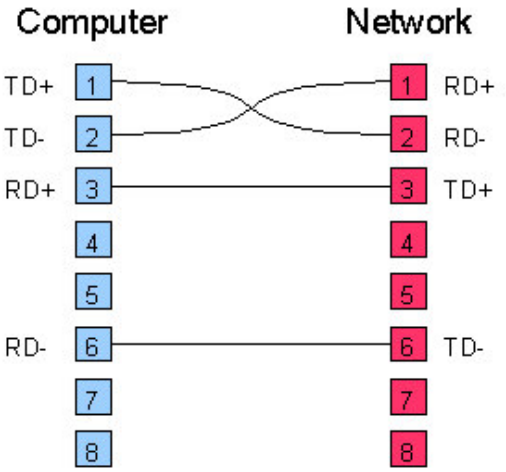

Figure B-3: Another design that works in some cases.

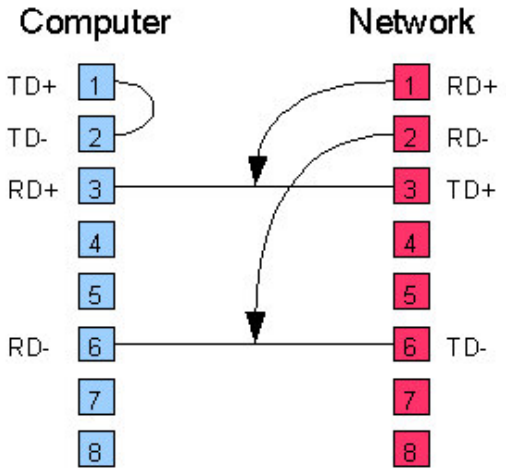

Figure B-4: A common one-way cable design, but may not be the best idea for an IDS.

way cabling claim that this works, though I am not very fond of this design. Here, in order to maintain connection on both ends, you loop the two transmission pins on the computer side together, and on the other end, you connect the two receiving pins to the network sending pins. The connection is maintained because the network receive pins see traffic, so they are happy. The reason I don't particularly like this design is because an Intrusion Detection System on a busy network should be flooded with traffic, and by patching the the network receive wires to the network send wires just to maintain a connection you inadvertently flood all traffic back to the network.

One final cable design is shown in figure B-5. This one is much more complicated and difficult to create, though it should be free of the problems in either of the other one-way cable designs. In this design, you again loop the two transmission lines on the computer side together to maintain an active link there. On the network side, however, you wire the two receive pins to a third port's transmission lines. This way, the third port will send active link requests (or in the case of a hub, any traffic on the line). This will trick the network link to thinking there is an active send-receive line available.

More resources for one-way cabling can be found on the Internet.

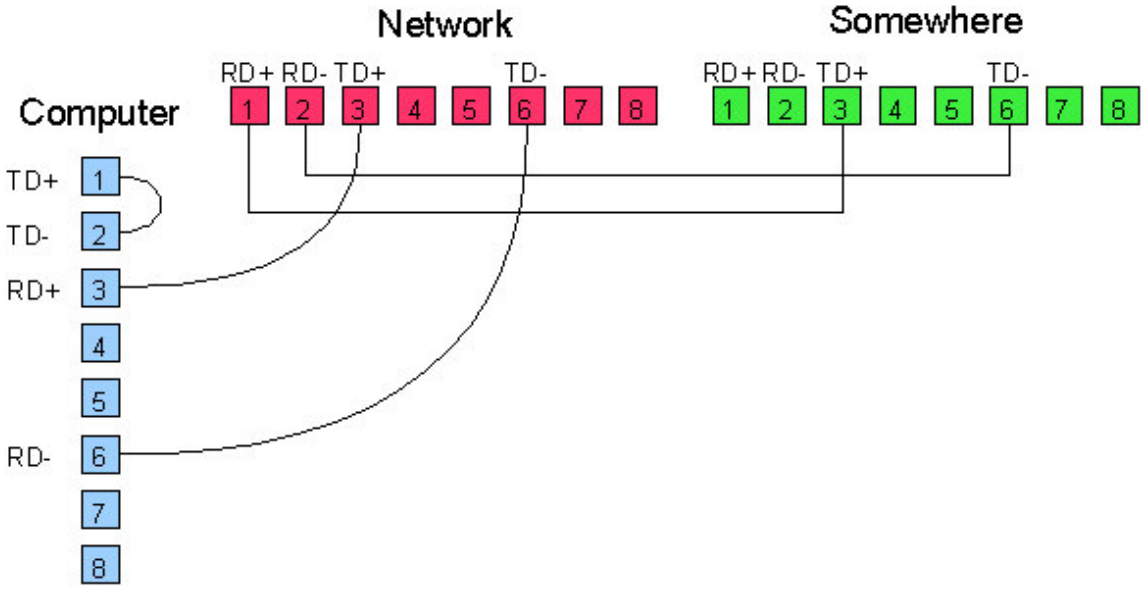

Figure B-5: A more complex one-way cable design.

Appendix C – Installing Snort and MySQL

The following is a step-by-step guide to install Snort with MySQL support on Debian Linux. We will be installing a version of Snort that is considered to be "testing" software, not "stable", so I assume you have prepared apt to allow you to install such programs.

It is important to note that this install procedure assumes you have not previously installed any other version of Snort or MySQL. If there was a prior version on the system, some configuration files may remain left over. In this case you should replace your old configuration files with the new ones included in the current installation. This is especially true for Snort since it has been through some major revisions in recent versions, and an old configuration file will cause chaos when you try running a new version.

1. Log in to Debian, open a terminal, and gain root access by running `su`. Root access will be required for many of the operations involved in the setup process, so you may need to do this step for any other terminals you open as you follow the steps of this guide.

2. When we install snort-mysql it will try to connect to MySQL, so we will first install and configure MySQL. Run the command:

   ```
   apt-get -t testing update
   ```

   This will tell apt to connect to all sites in your `sources.list` file and get lists of available packages or updates.

3. Now, run the command:

   ```
   apt-get -t testing install mysql-server
   ```

   The output from the command should look similar to the following:

   ```
   Reading Package Lists... Done
   Building Dependency Tree... Done
   The following extra packages will be installed:
     gawk libdbd-mysql-perl libdbi-perl libmysqlclient12
     libnet-daemon-perl libplrpc-perl mysql-client mysql-
     common
   Suggested packages:
     dbishell libcompress-zlib-perl mysql-doc
   The following NEW packages will be installed:
     gawk libdbd-mysql-perl libdbi-perl libmysqlclient12
     libnet-daemon-perl libplrpc-perl mysql-client mysql-
     common mysql-server
   0 upgraded, 9 newly installed, 0 to remove and 0 not
   upgraded.
   Need to get 6192kB of archives.
   After unpacking 15.0MB of additional disk space will be
   used.
   Do you want to continue? [Y/n]
   ```

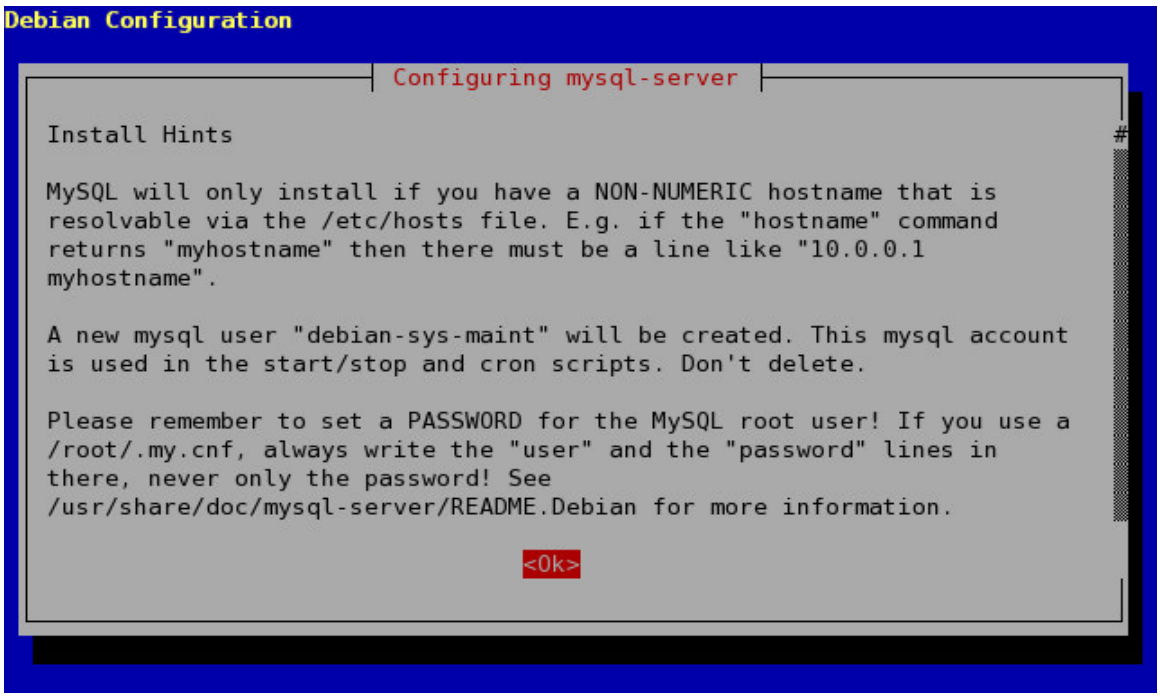

Figure C-1: Installing MySQL

As you can see, MySQL comes in several parts, namely mysql-server, mysql-client, and mysql-common. These components all depend on one another, and since apt resolves dependency issues automatically, it will install any necessary components. Press enter to continue with the installation.

4. Apt will now download, unpack, and install all required packages. There will only be one interactive step to the installation. You will be shown the information in figure C-1. The most important part of this message is the reminder to set a root password.

5. Run either `pstree` or `ps -A` to ensure that `mysqld`, the MySQL server (daemon) is running. If it is not, either start it by typing `mysqld`, or by restarting your computer.

6. If `mysqld` is running, type `mysql` to run the MySQL client. No user name or password is needed yet since we will now set the root password as well as create a user account that Snort will use to connect. You should be greeted and prompted by MySQL as follows:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version:
4.0.23_Debian-7-log

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

mysql>
```

7. Type the following command to set your root password:

```
SET PASSWORD = PASSWORD('my_password');
```

Where `my_password` is a placeholder for your actual password. If you enter the command correctly, MySQL will respond with something like:

```
Query OK, 0 rows affected (0.00 sec)
```

8. Now to test your root password, type `QUIT` to return to your shell. This time, you must run `mysql -p` to tell it you want to enter a password. You will be prompted to enter the password you just created. Type it and press enter to return to the MySQL prompt.

9. Now we will create a database in which Snort can log it's data. Once you have logged in to MySQL as root, enter the following command:

```
CREATE DATABASE snort;
```

You should be given the following reply to indicate success:

```
Query OK, 1 row affected (0.00 sec)
```

10. We will now create a user name and password for Snort to use to connect to MySQL. It is in general a bad idea to allow Snort to use the root account, especially if the MySQL server Snort is connected to holds additional information. The user account will be given only the minimum permissions required for operation. To do this, enter the following command:

```
GRANT INSERT, SELECT ON snort.* TO 'snort'@'localhost'
IDENTIFIED BY 'my_password';
```

You should replace “my_password” with a password of your choosing. This command grants privileges to a local user. If you wish to set up Snort to connect to a remote MySQL database, run the command on the computer with the database, and replace “localhost” with the name of the computer that can connect by this user, or use the wild card % to mean that the user “snort” can connect from any computer. To test the new account, type `QUIT` to exit the MySQL client. At the shell prompt, type:

```
mysql --user=snort -p
```

you will be prompted for the password you selected when you ran the GRANT query. Type it, press enter, and you should be back at the MySQL prompt.

11. We are finally ready to install Snort. If you haven't already done so, type QUIT to leave the MySQL prompt and return to your shell. Run the following command:

```
apt-get -t testing install snort-mysql
```

The initial output should look something like the following:

```
Reading Package Lists... Done
Building Dependency Tree... Done
```

```
The following extra packages will be installed:
  libpcap0.8 snort-common snort-rules-default
Suggested packages:
  snort-doc
Recommended packages:
  oinkmaster
The following NEW packages will be installed:
  libpcap0.8 snort-common snort-mysql snort-rules-
  default
0 upgraded, 4 newly installed, 0 to remove and 89 not
upgraded.
Need to get 797kB of archives.
After unpacking 3076kB of additional disk space will be
used.
Do you want to continue? [Y/n]
```

As you can see, apt will automatically install snort-common and snort-rules-default for you. Press enter to confirm the installation.

12. You will now be presented with a series of interactive steps to configure Snort. This first is simply an information window shown in figure C-2. You can read the information provided and press enter to continue.

13. The second step asks you to identify the network interface you want Snort to listen on, as shown in figure C-3. The default value is eth0, however if you have changed your network configuration, or have multiple network cards in your computer you may want to specify a different interface. By opening up another shell and running the command `ifconfig` you can see a list of the available interfaces. Once a device is selected, press

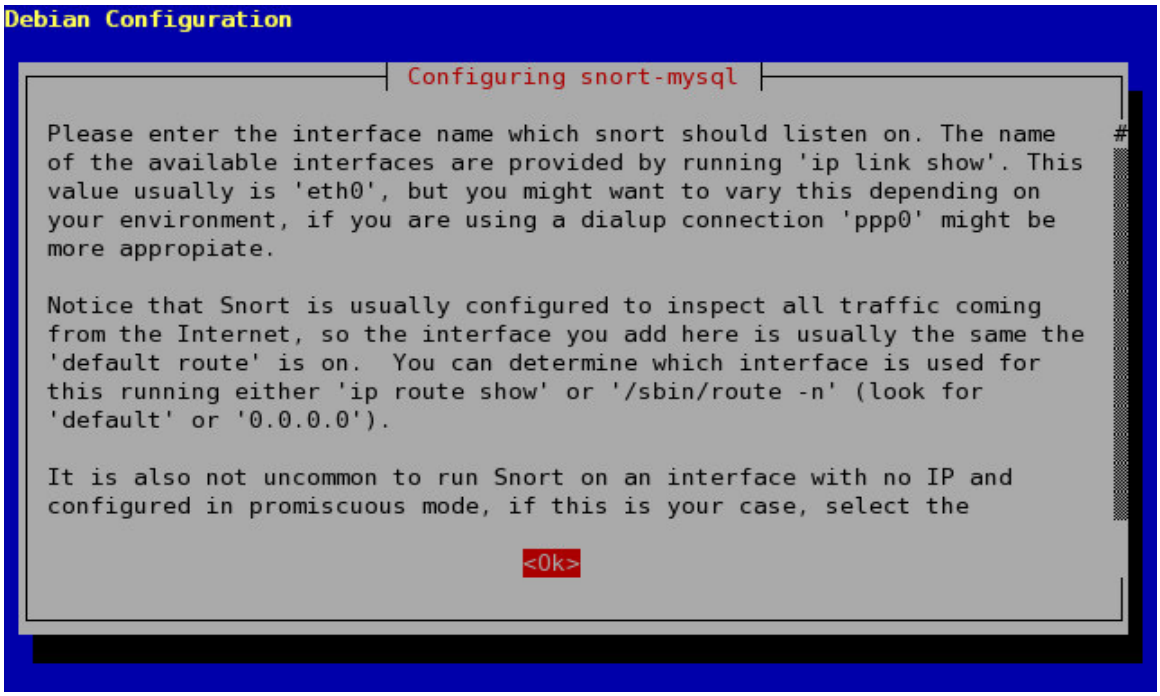

Figure C-2: The Snort interactive setup.

Figure C-3: Identify the listening interface.

enter to continue.

14. The third step is to identify what address Snort will listen to for potential threats. Addresses must be listed in CIDR format (ie: the default value is a CIDR value of 192.168.0.0/16), and multiple ranges can be specified by separating them by commas with no spaces. To listen for threats on any address type 'any' as shown in figure C-4. This will provide the safest, most comprehensive scan, since IP addresses can be spoofed.

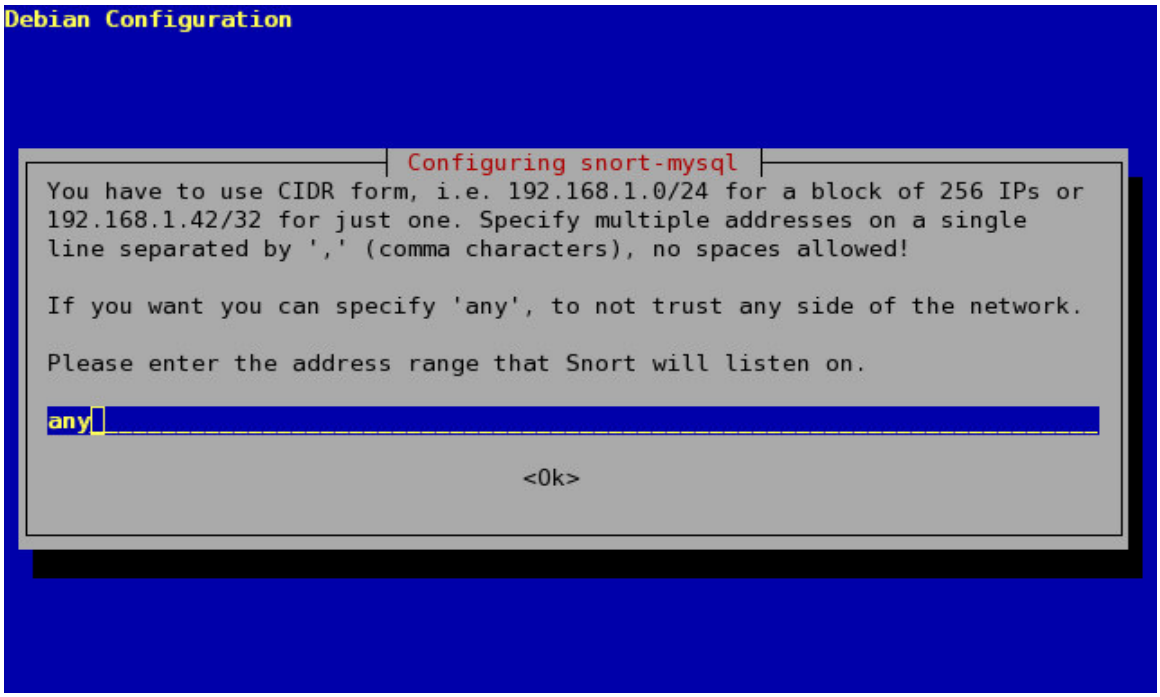

Figure C-4: Identify the address range for listening.

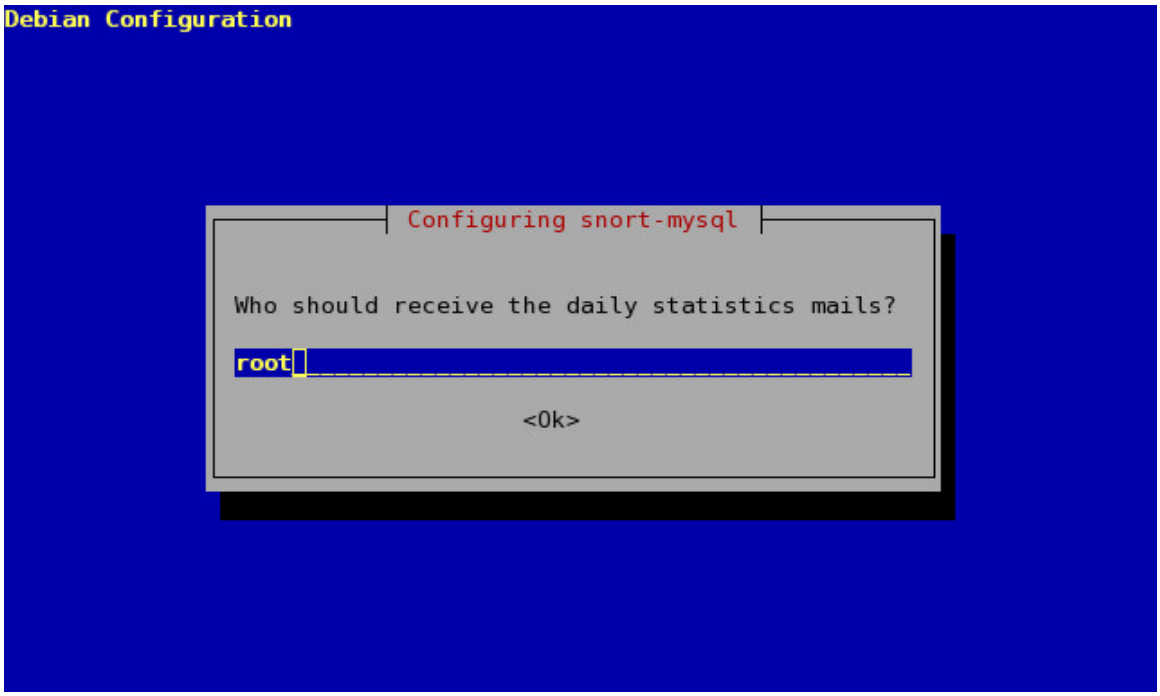

Figure C-5: Configuring daily statistics mail.

Once you have selected your target addresses, press enter to continue.

15. The next screen, shown in figure C-5, allows you to give Snort an e-mail address to send daily statistical e-mails to. You must have mail delivery configured to use this option.

16. Snort setup will now ask you if you are prepared to configure MySQL access, as shown in figure C-6. Since we have already installed MySQL, created a database, and granted a

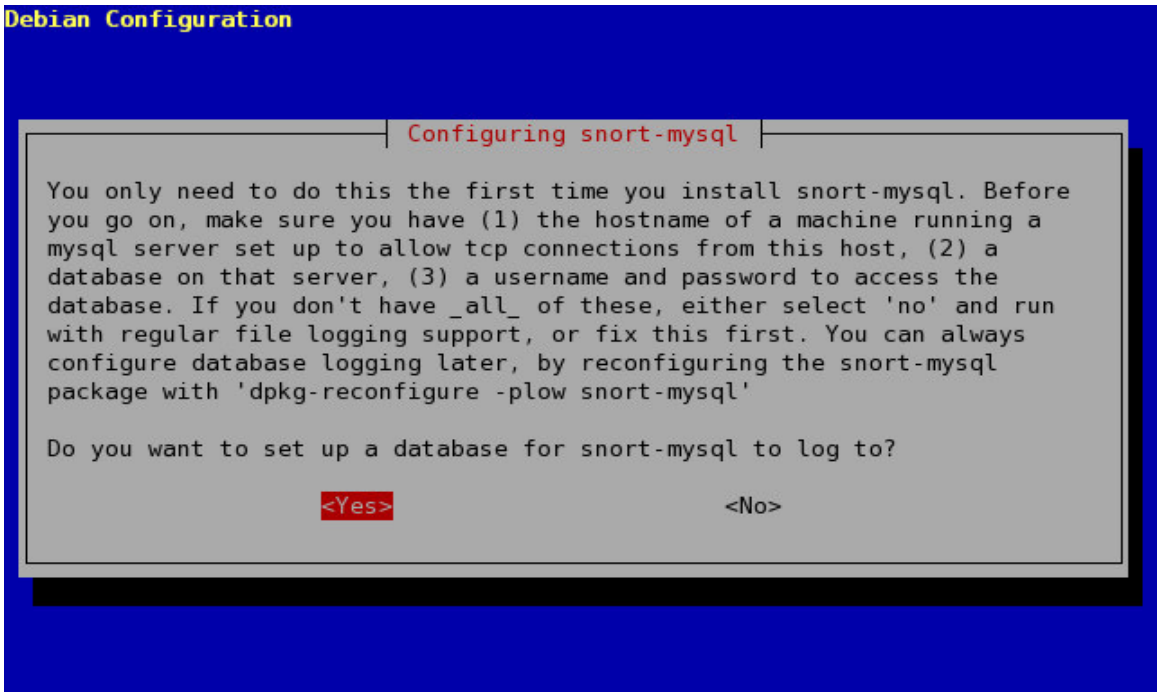

Figure C-6: Preparing to configure MySQL access.

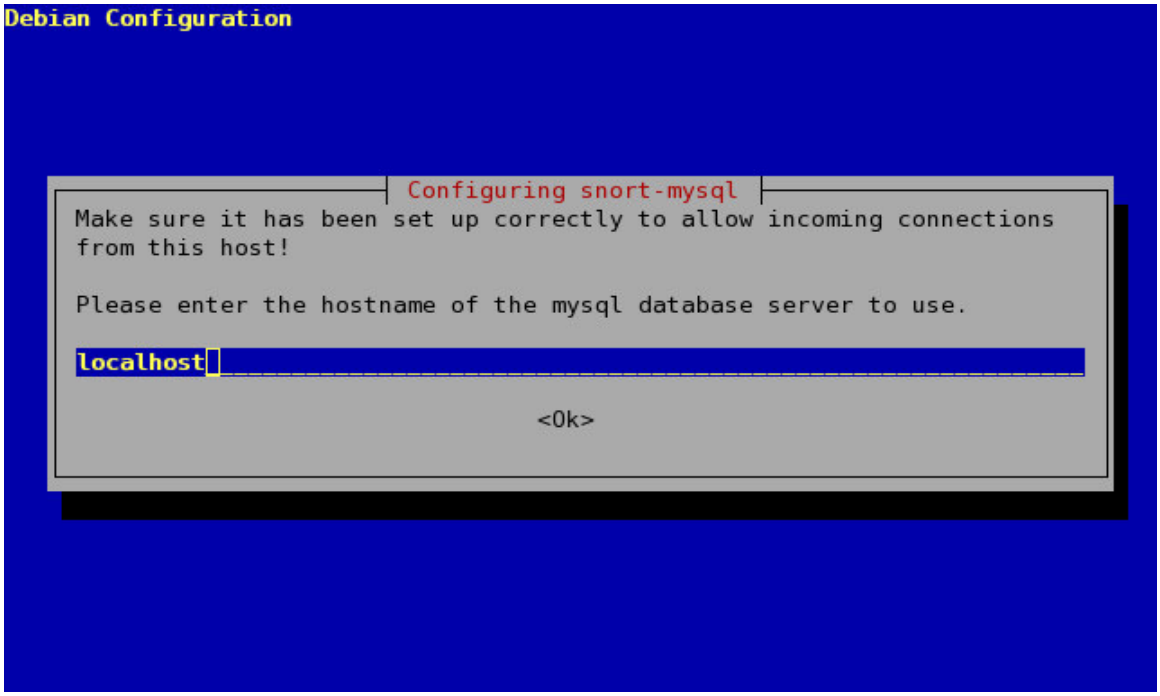

Figure C-7: Connecting to the MySQL server.

user account access, we can say "yes" and set up snort-mysql with the configuration utility. Press enter to continue.

17. The first MySQL-related question is the location of the server containing the database that Snort will log its data to. If you have followed this guide completely and installed MySQL on the same machine as Snort, you can enter "localhost" as shown in figure C-7. Otherwise, type the name of the computer that has is running your MySQL server.

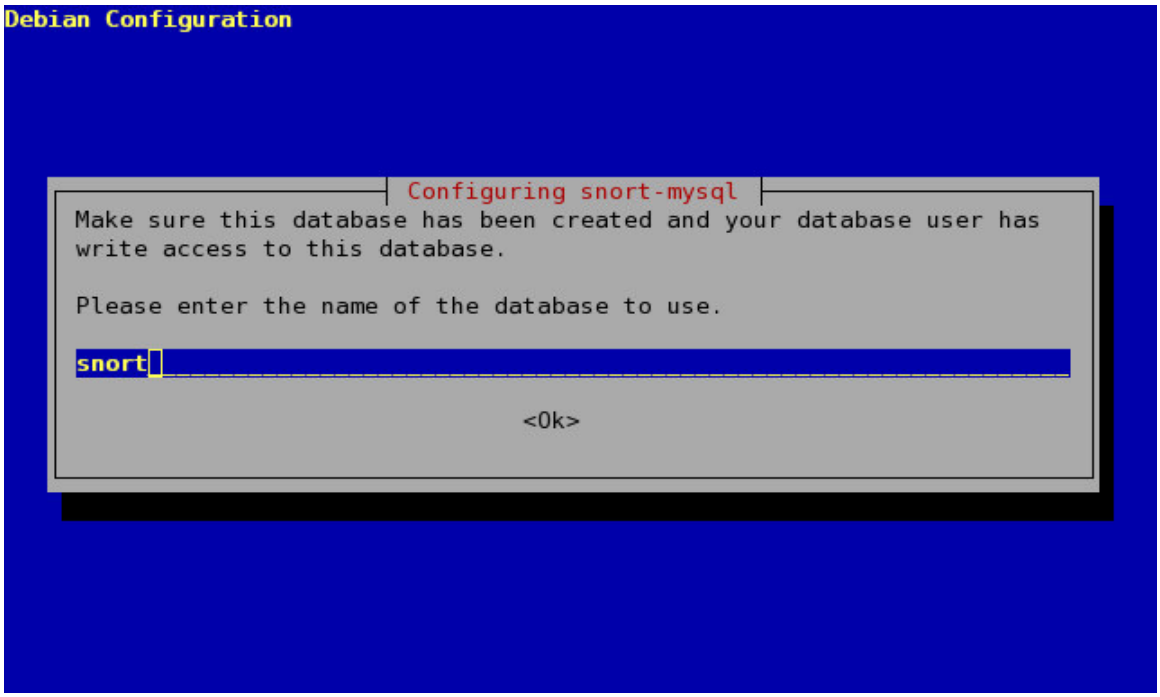

Figure C-8: Selecting a MySQL database for use by Snort.

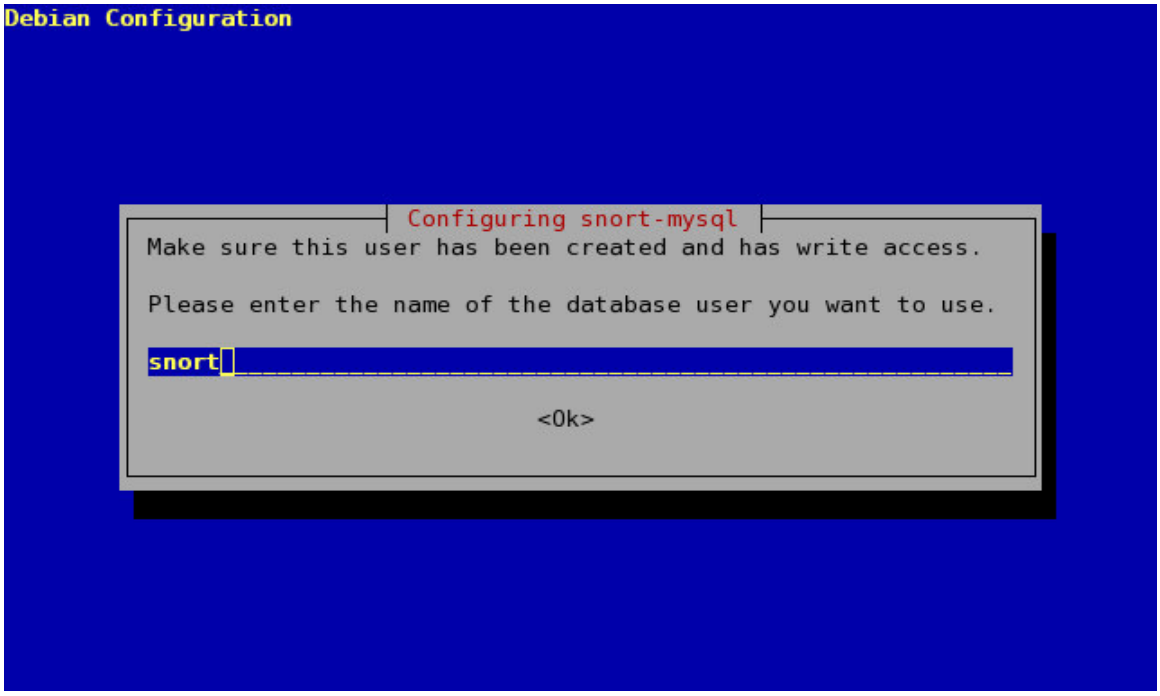

Figure C-9: Giving Snort its MySQL account information.

18. The configuration program now asks for the name of the database that Snort will log to as shown in figure C-8. This database should be created prior to running this part of the configuration script. The database in question is the one we created in step 9 of this guide. Type the name you gave the database. The one we used in this guide was “snort”.

19. Next, Snort asks for its MySQL account information. Type the name of the account that you created for Snort in step 10 of this guide. If you accepted the name used here, then

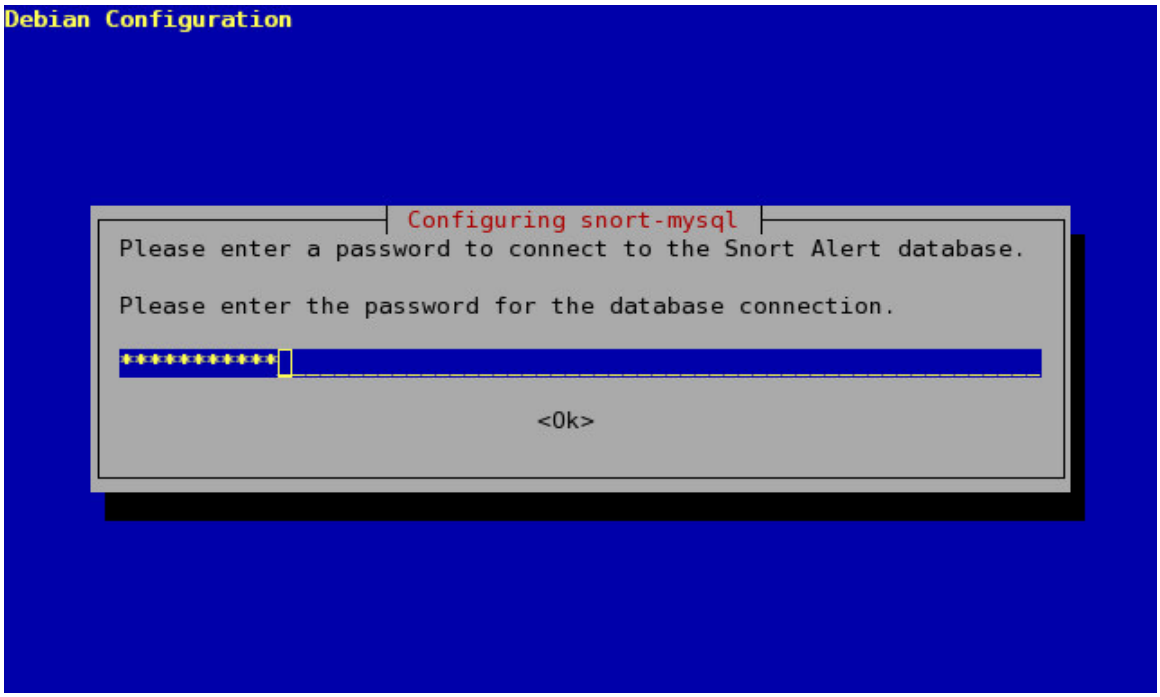

Figure C-10: Enter your password.

enter the user name “snort” as shown in figure C-9.

20. The next prompt, shown in figure C-10, requests the password associated with the account given in the previous step.

21. The final step for this part of the setup process is shown in figure C-11. This information screen tells you that you must now set up the database structure that Snort requires in order to run. Snort requires 16 tables with special structures for purposes of logging and storing data. The installation package comes with an SQL query file that will create all these tables for you. The problem, however, is that the configuration utility runs before files are actually unpacked, so the file referenced here does not exist in that location yet. There are a couple solutions to this problem:

    1) One solution is to extract all files from the package and follow the instructions given. To do this, first open a new terminal and locate the snort-mysql Debian package downloaded by apt. It should be located in `/var/cache/apt/archives/` as `snort-mysql_2.3.2-1_i386.deb`, or something similar depending on the version and architecture you are using. If the package is not in this location for some reason, you can find it by running the following:

    ```
    find / -name snort\*.deb
    ```

    Once you have located the package, run the following commands:

    ```
    cd /var/cache/apt/archives/
    cp snort-mysql_2.3.2-1_i386.deb /tmp/
    ```

Figure C-11: The final step in the interactive installer.

Replace the name of the file with the one you located if it is different. This will copy the package to the `tmp` directory. Now do the following:

```
cd /tmp/
dpkg -x snort-mysql_2.3.2-1_i386.deb snorttemp
```

This will extract all files in the package to the temporary directory `snorttemp`. Now we will follow the direction given by the message and set up the database tables by running:

```
cd snorttemp/usr/share/doc/snort-mysql/
zcat create_mysql.gz | mysql -u root -p snort
```

You will need to perform this operation as root, since snort was only given insert and select privileges. Also, if you followed this guide and installed MySQL on the same machine as Snort, the "-h <host>" part of guideline command from the information window is unnecessary. After you run this, you will be prompted for the root password. If the operation runs successfully, you will not be given any message to indicate so. If you would like to make sure the operation was successful, open a MySQL client, and run the following commands:

```
USE snort;
SHOW TABLES;
```

This will give you a list of all tables in the snort database. You can now delete the temporary files created to perform this operation by running the following at the shell prompt:

```
cd /tmp/
rm snort-mysql_2.3.2-1_i386.deb
rm -r snorttemp
```

Then leave your terminal open and continues to the next step.

2) This is not the only option available at this point. If you do not want to make a copy of the package from your cache files and extract it, you can simply press enter to continue without having set up the database structure. This is the last step in the configuration utility regardless of whether you complete it or not. When setup completes, it will try to run Snort in the background and if you have not configured everything properly it will simply fail to launch. At this point you can carry out the instructions given by the information screen and run:

```
cd /usr/share/doc/snort-mysql/
zcat create_mysql.gz | mysql -u root -p snort
```

As stated above, the "-h <host>" part is not required when the MySQL server is on the local machine.

22. If you remember when we made the MySQL user account `snort` for Snort to use to access its database, we gave it insert and select privileges on the entire database. Now that the tables have been created, there is one more privilege the account needs. Log into MySQL by typing:

```
mysql -u root -p
```

Enter your root password to arrive at the MySQL prompt. Then type:

```
GRANT INSERT, SELECT, UPDATE ON snort.sensor TO
'snort'@'localhost';
```

Snort uses the sensor table to keep track of some run-time data, and needs to be able to change the information in this table. Again, adapt this line to match your situation if your MySQL server is on a remote computer. Now your configuration is complete and Snort may now be running. If it is not running, it should run properly the next time it is started. To start Snort you can either reboot or start it manually.

If you make a mistake at any point during the configuration process, don't panic. You can always access the configuration utility again by running the following command:

```
dpkg-reconfigure snort-mysql
```

This will restart the configuration utility for you. This version of the utility is, for the most part, the same as the one ran at install time. The only difference is that there are several additional questions asked, but you can simply accept the default values as they are good for any standard usage of Snort.

Appendix D – Installing Acidlab and Apache-SSL

The following is a step-by-step guide to install and secure Acidlab for purposes of managing Snort alerts. Acidlab is a PHP script, so we will also need a web server and the PHP processing engine in order to run it. Since the information Acidlab deals with is potential security threats on your network, it is important that the information is not seen by any unauthorized people so we will be using a secure web server. To get Acidlab running we will be installing Apache-SSL secure web server and the PHP engine.

Again, this guide assumes there is no prior version of any of these applications installed on your system. If there is, and you wish to follow this guide to setting up an IDS, it is recommended you replace any old configuration files when asked. Otherwise, a knowledge of Apache configuration will be required to complete the setup process.

1. First open up a terminal and gain root access by running `su` and typing your root password. As in the previous section, root access will be required for many of the operations in this guide.

2. Acidlab is a PHP web site so we first need to install a web server and the PHP engine. We will now install the Apache-SSL secure web server. Run the command:

   ```
   apt-get -t testing update
   ```

   To update package lists.

3. Run the following command:

   ```
   apt-get -t testing install apache-ssl
   ```

   This will commence installing Apache-SSL. You should get output similar to the following:

   ```
   Reading Package Lists... Done
   Building Dependency Tree... Done
   The following extra packages will be installed:
     apache-common apache2-utils libapr0 openssl ssl-cert
   Suggested packages:
     apache apache-perl apache-doc ca-certificates
   The following NEW packages will be installed:
     apache-common apache-ssl apache2-utils libapr0 openssl
     ssl-cert
   0 upgraded, 6 newly installed, 0 to remove and 16 not
   upgraded.
   Need to get 2462kB of archives.
   After unpacking 6718kB of additional disk space will be
   used.
   Do you want to continue? [Y/n]
   ```

   As you can see, several other packages are required by Apache-SSL. Notably,

OpenSSL is the program we will use to generate and sign a certificate. Press enter to start the installation.

4. You will now be asked a series of questions by two configuration utilities. The first is to configure Apache-SSL, and there is only one question, shown in figure D-1. This deals with CGI scripts, which we won't be using anyway, so the default answer of “No” will be fine.

5. The second configuration utility is for OpenSSL. It will ask you a series of questions in order to generate a certificate for you. Since this procedure is very basic, there are no pictures of this included, but provided below are a list of the information OpenSSL will gather:

   1) A two-letter country code (ie: CA).
   2) Your State or Province name.
   3) Your city or town name.
   4) You organization's name.
   5) Your division or section within the organization.
   6) The host name of the server for which the certificate is being generated.
   7) An e-mail address to be associated with the certificate.

6. The installation process will complete and you now have a fully operating web server using SSL with 128-bit encryption and a self-signed certificate. If you would like to ensure that this part of the setup process completed correctly, open a web browser and

Figure D-1: Setting up Apache-SSL

type the location `https://localhost`. If you make a mistake at any point you can access either of these configuration tools again by running either of these commands:

```
dpkg-reconfigure apache-ssl
dpkg-reconfigure openssl
```

7. Now we will install the PHP processor. Run the following command:

```
apt-get -t testing install php4
```

You should see output similar to the following:

```
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  libapache-mod-php4 libzzip-0-12 php4-common
Suggested packages:
  php4-pear
The following NEW packages will be installed:
  libapache-mod-php4 libzzip-0-12 php4 php4-common
0 upgraded, 4 newly installed, 0 to remove and 16 not
upgraded.
Need to get 1842kB of archives.
After unpacking 3674kB of additional disk space will be
used.
Do you want to continue? [Y/n]
```

Press enter to continue.

8. The setup process will automatically update the `/etc/apache-ssl/modules.conf` file for you to contain the line to load the PHP module. Apache must be restarted before it will recognize the new module. To restart your Apache web server, use the following command:

```
apache-sslctl restart
```

If you wish to test the PHP engine to make sure everything is working correctly, enter the following commands:

```
cd /var/www
echo “<?php phpinfo(); ?>” > test.php
```

Then open up a web browser and type the URL `https://localhost/text.php`. This will display the PHP engine and web server information. After you have confirmed that PHP is working, make sure to remove the `test.php` file.

9. Since Acidlab accesses MySQL, it needs an user name and password. Acidlab allows management of Snort alerts, and therefore requires more privileges to operate than Snort

itself needs, so we will create a second account for Acidlab to use. Type:

```
mysql -u root -p
```

Followed by your root password to acquire a MySQL prompt. At the prompt, type:

```
GRANT CREATE, INSERT, SELECT, UPDATE, DELETE ON snort.* TO
'acid'@'localhost' IDENTIFIED BY 'my_password';
```

Acidlab supports the feature of archiving old alerts. After alerts have been analyzed, you may not want them to mix in with new alerts. A simple solution is to delete them, however once this is done they are unrecoverable. If you don't want to permanently remove them, but you don't want them interfering with new analysis, you can archive them. If you would like to be able to use this feature, you must now create a database with the same structure as Snort's database for archiving. To do this, first create an archive database from the MySQL prompt as follows:

```
CREATE DATABASE snort_archive;
GRANT CREATE, INSERT, SELECT, UPDATE, DELETE ON
snort_archive.* TO 'acid'@'localhost';
```

Now, quit MySQL and do type the following commands:

```
cd /usr/share/doc/snort-mysql/
zcat create_mysql.gz | mysql -u root -p snort_archive
```

10. We are now ready to install Acidlab. There are several versions of Acidlab for different database systems. The default is PostgreSQL if we simply tell apt that we want the `acidlab` package. We need to install the `acidlab-mysql` package by entering the following command:

```
apt-get -t testing install acidlab-mysql
```

You should see initial output similar to the following:

```
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  acidlab libgd2-xpm libphp-adodb libphp-phplot libt1-5
  php4-gd php4-mysql wwwconfig-common
Suggested packages:
  libgd-tools postgresql-client apache
Recommended packages:
  php4-pgsql php4-sybase php4-odbc
The following NEW packages will be installed:
  acidlab acidlab-mysql libgd2-xpm libphp-adodb libphp-
  phplot libt1-5 php4-gd php4-mysql wwwconfig-common
0 upgraded, 9 newly installed, 0 to remove and 16 not
upgraded.
```

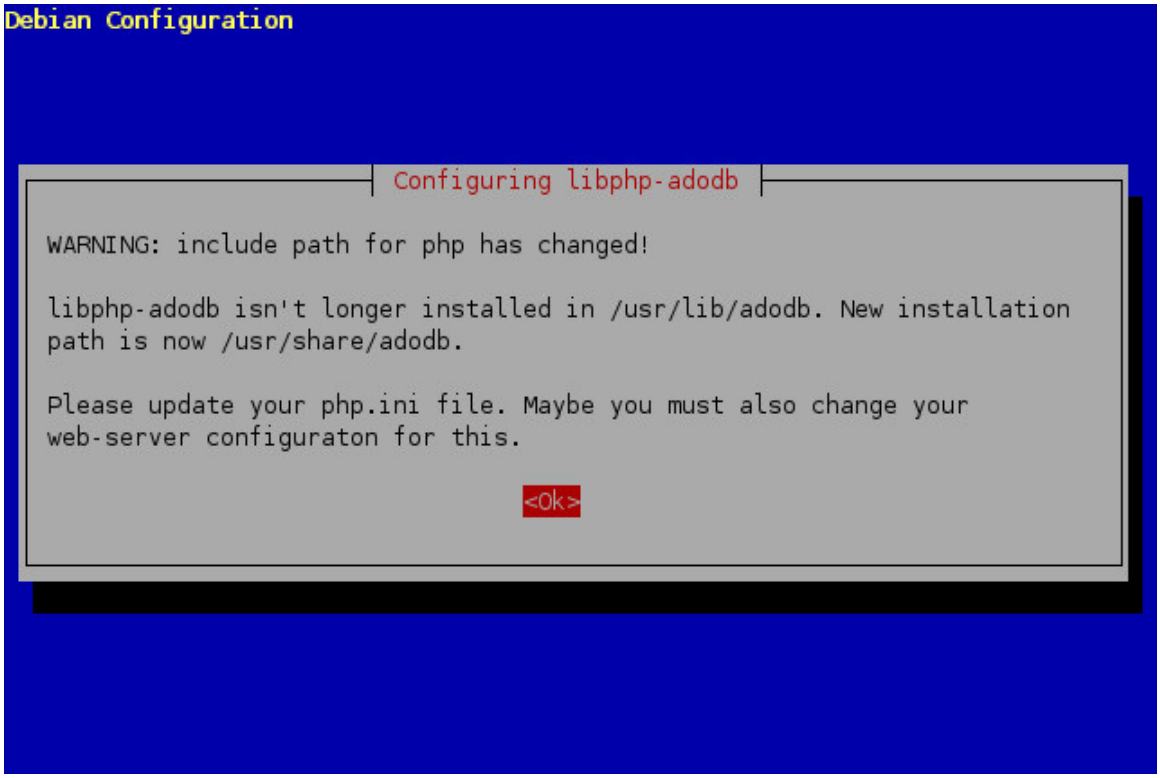

Figure D-2: Some information about PHP updates.

```
Need to get 1617kB/1639kB of archives.
After unpacking 5087kB of additional disk space will   be
used.
Do you want to continue? [Y/n]
```

Press enter to continue.

12. The configuration utility will once again launch. You should first see the information window shown in figure D-2. ADOdb is a library that abstracts the interface to many common database servers. We are only using one database, MySQL, so this information is not critical for our application. Press enter to continue.

13. You will now be given the option to have your web server configuration file updated automatically. Simply select Apache-SSL as shown in figure D-3.

14. The next step is to tell Acidlab what type of database it will be working with, as you can see in figure D-4. Select MySQL, and press enter to continue.

15. You will now be asked a series of straight-forward questions about database access. The following is a list of the information the Acidlab configuration utility will ask you for:

    1) The name of the database that Snort uses for logging alerts. In our case, this is `snort`.
    2) The computer on which the database server resides. This should be `localhost`.
    3) The user name Acidlab will use to access this database. This is name we created in step 9: `acid`.

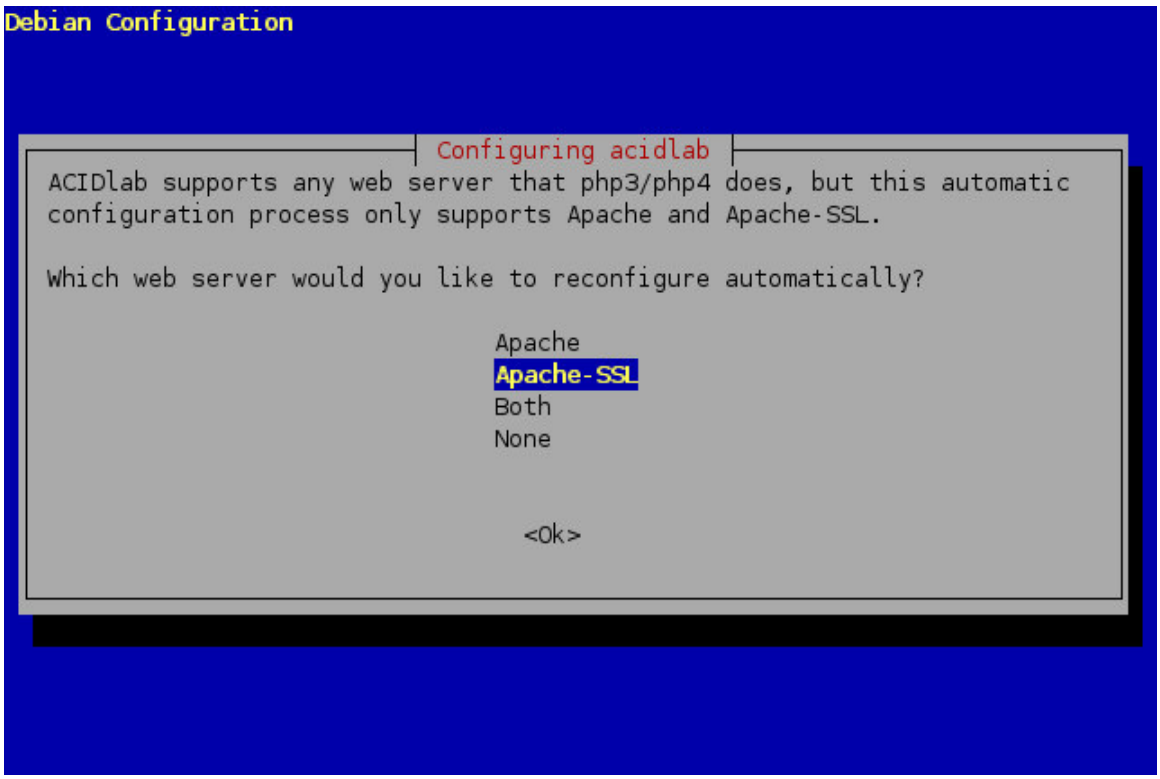

Figure D-3: Select the web server you want to have reconfigured.

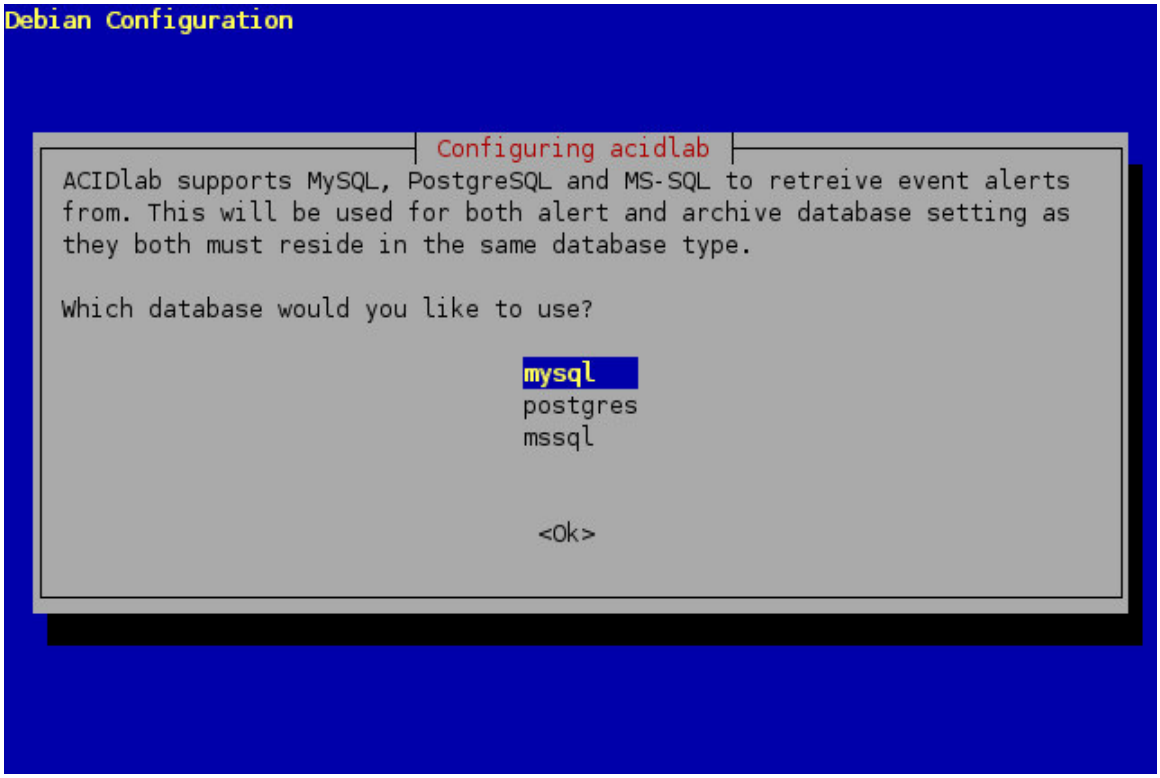

Figure D-4: Tell Acidlab what type of database it will be working with.

4) The password you associated with this account.
5) The name of the database you created to be used as an archive. We called it `snort_archive`.
6) The name of the computer storing this database. This, again, is `localhost`.
7) The user name Acidlab can use with this database. Again, this is `acid`.
8) The password you associated with this account.

16. Once you have entered all of the above information, you should see the information screen shown in figure D-5. This is a note about how to finish configuring Acidlab. We will do what it says shortly, There is one other thing to complete first. Press enter to continue.

17. Setup will now complete, and you should be returned to the command prompt. At this point, most things are ready to go, except the PHP configuration file may not have been updated for MySQL support. Change directory to `/etc/php4/apache/` and edit the file `php.ini`. Search for a line like the following:

```
;extension=mysql.so
```

This should appear in the configuration block titled "Dynamic Extensions". Once you have located it, uncomment it by delete the semi-colon in front of the line, so your line looks like this:

```
extension=mysql.so
```

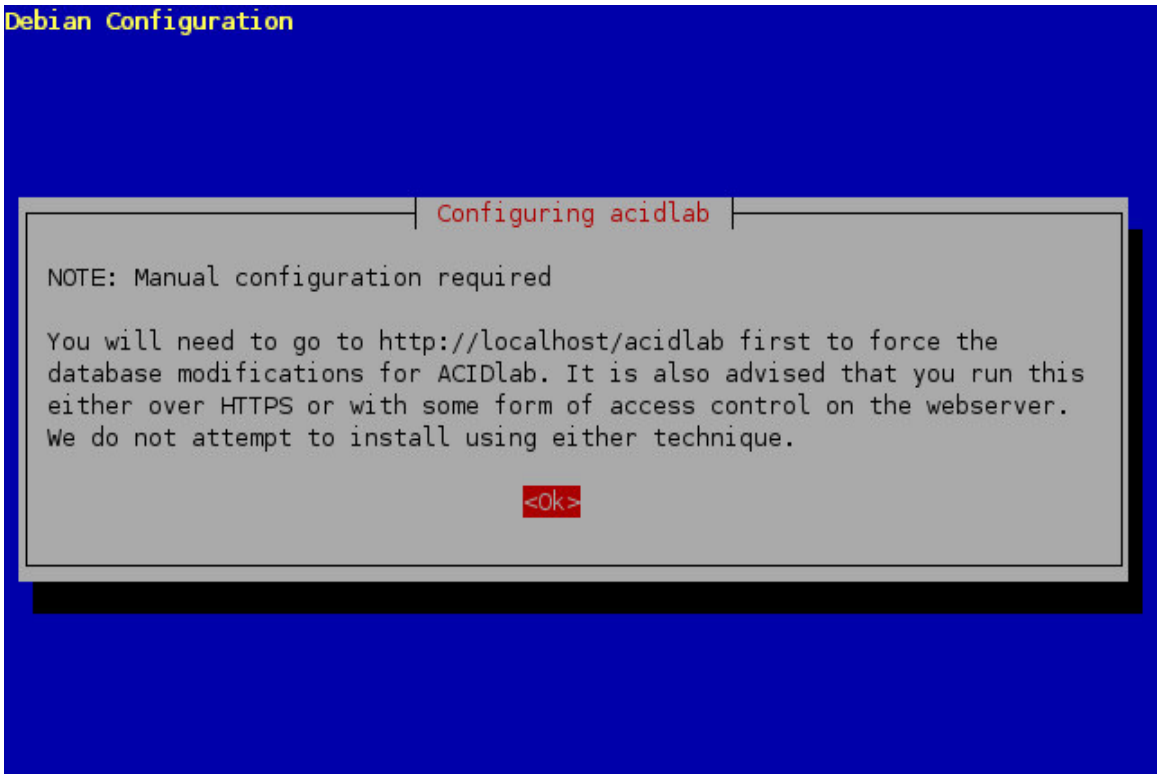

Figure D-5: A note about configuring Acidlab.

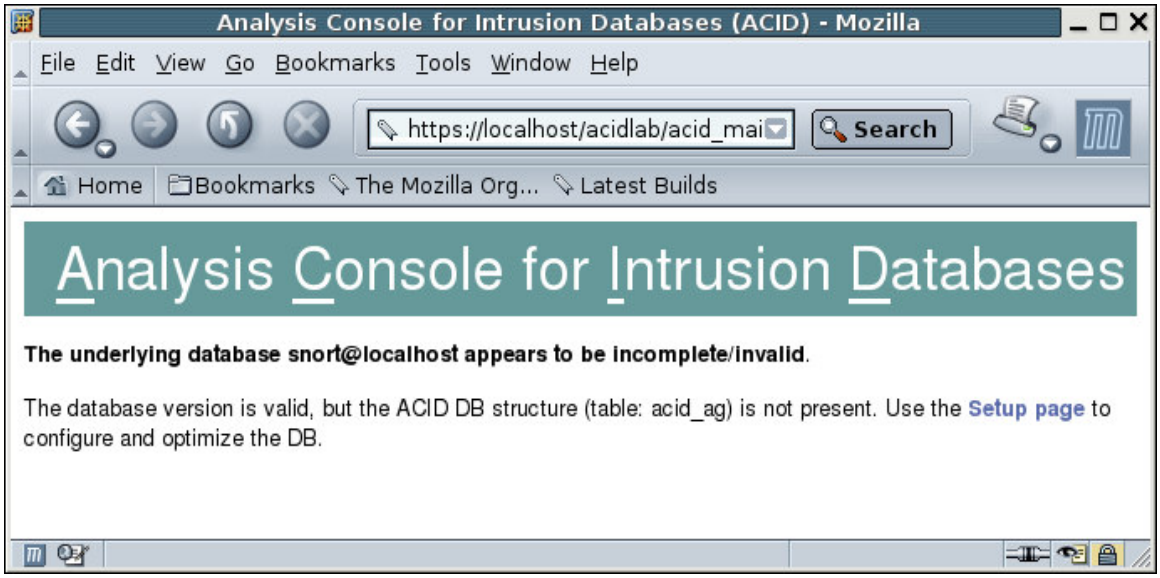

Figure D-6: Acidlab still needs to be set up.

If the line does not exist in your configuration file, make sure you include it. Now save the file and close your editor.

18. Before we follow the instructions given to us in figure D-5, we need to restart the Apache web server so that it reads the new configuration files. To restart Apache, type the following:

```
apache-sslctl restart
```

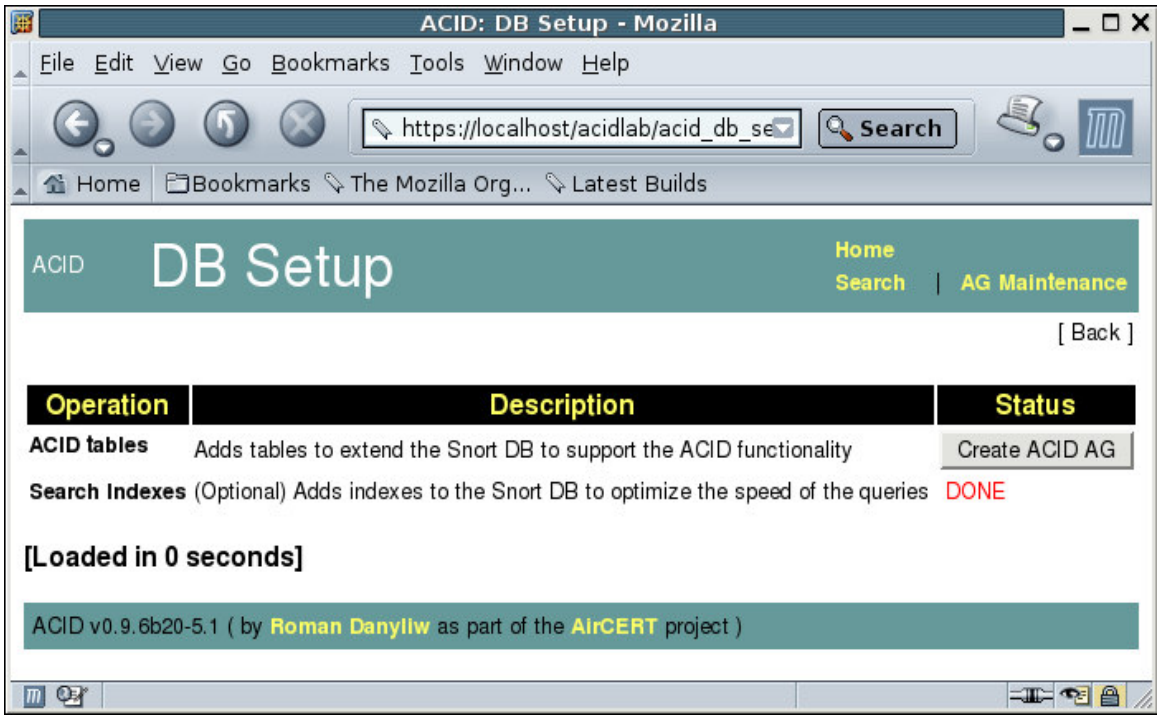

Figure D-7: The Acidlab setup page.

19. You should now be able to open a web browser and type the address `https://localhost/acidlab` to access Acidlab. You should see a page like that shown in figure D-6. You can't access Acidlab just yet, there is still some setup left to do. Click the link titled “Setup page” to continue.

20. You should see a page like the one in figure D-7. This page is used to create database tables for Acidlab. Acidlab uses four tables in Snort's alert database to store application data. To create these tables, simply click the button labeled “Create ACID AG”.

21. After clicking the button, the four tables will be created, and you should see the page shown in figure D-8. Notice the four red lines at the top, telling you that the four tables

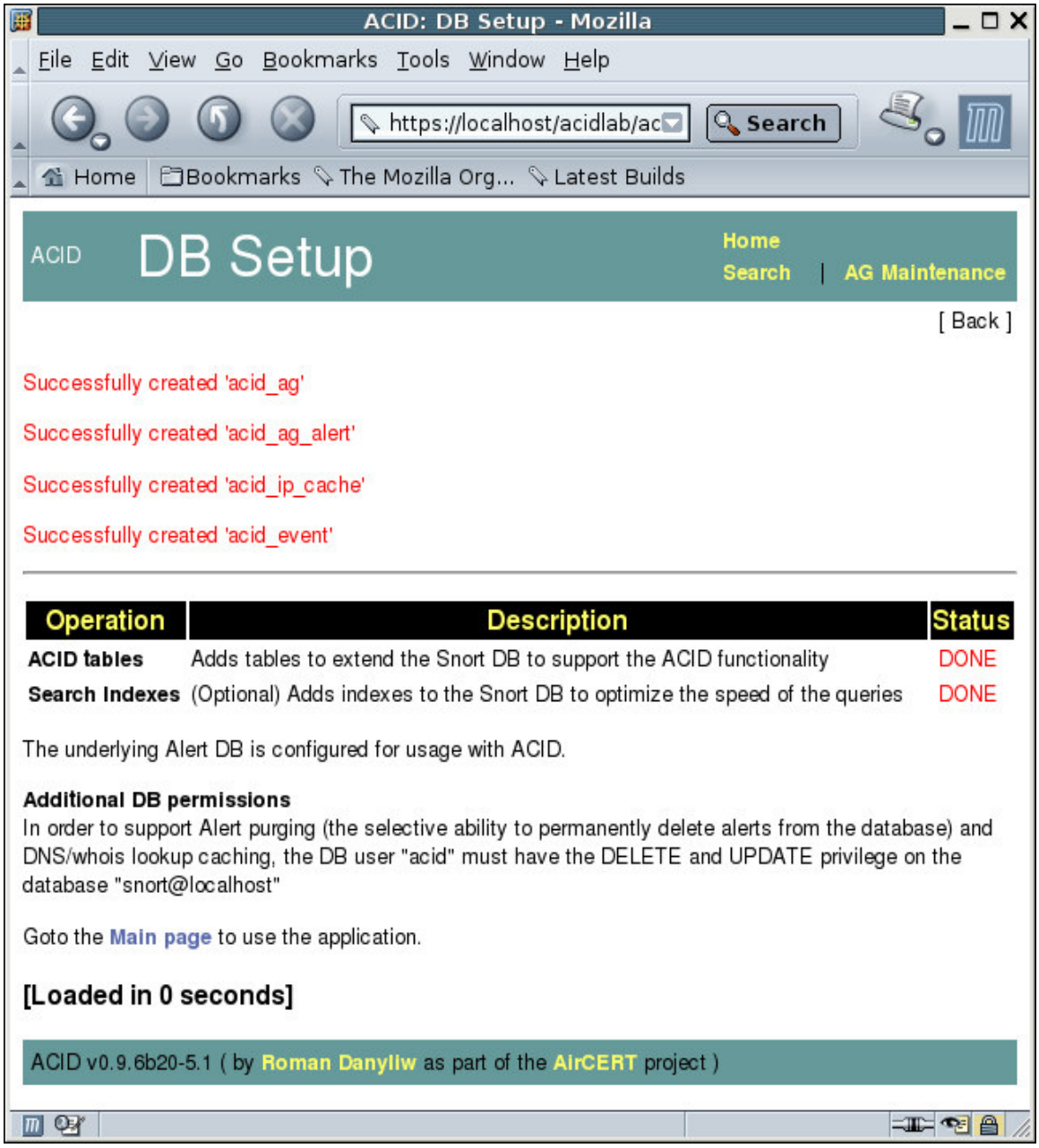

Figure D-8: Tables created successfully.

were created successfully. Also notice all operations in the list are flagged as done and there is now a link at the bottom of the page inviting you to go to the main page. You can now click this link to finally open Acidlab.

22. After you have taken a look at your newly installed program, there is one more thing to be done. At the moment, communication with Acidlab is secure, but it is not restricted. In other words, data sent by Acidlab will be encrypted, but anyone can access it. We will now configure Apache-SSL to require the use of a user name and password to access Acidlab. The first step is to create a file that contains a user name and password that Apache will use to verify incoming requests. Return to your terminal and type the following:

```
htpasswd -c /etc/apache-ssl/password snort
```

You may replace `snort` with a user name of your choice. You will then be prompted twice for a new password for this account. After which you should get the following message to indicate the password file was created successfully:

```
Adding password for user snort
```

23. Now, change your current directory to `/etc/apache-ssl/` and type the following commands:

```
chown root.nogroup password
chmod 640 password
```

24. Now open up the file `/etc/apache-ssl/httpd.conf` in your favorite editor and locate the following block of text:

```
<Directory />
    Options SymLinksIfOwnerMatch
    AllowOverride None
</Directory>
```

Modify this block to look like the following:

```
<Directory />
    Options SymLinksIfOwnerMatch
    AllowOverride None
    SSLRequireSSL
    AuthType Basic
    AuthName "Administrators"
    AuthUserFile /etc/apache-ssl/password
    Require user snort
</Directory>
```

If you used a user name other than `snort`, make sure you replace `snort` in this block with the name that you picked.

25. There is one more additional step you can take to make sure your Apache-hosted site is only accessed by those you want. You can configure further access control by specifying what hosts can connect to the site. If you want to configure Acidlab to only be accessible on the local machine then type the following into your shell:

```
cd /etc/acidlab
```

Open up the file called `apache.conf` in your favorite editor. This file contains the configuration for the Acidlab directory and is referenced in the main Apache configuration file as an include. The default contents of the file are listed below:

```
Alias /acidlab  /usr/share/acidlab

<DirectoryMatch /usr/share/acidlab/>
Options +FollowSymLinks
AllowOverride None
order allow,deny
allow from all
<IfModule mod_php3.c>
  php3_magic_quotes_gpc Off
  php3_track_vars On
  php3_include_path .
</IfModule>
<IfModule mod_php4.c>
  php_flag magic_quotes_gpc Off
  php_flag track_vars On
  php_value include_path .
</IfModule>
</DirectoryMatch>
```

Acidlab installs itself to `/usr/share/acidlab`, not to your web root so it needs to be aliased. The `DirectoryMatch` block configures permissions for the directory. To configure access control, change these lines:

```
order allow,deny
allow from all
```

To say the following:

```
order deny,allow
deny from all
allow from host.domain.com
allow from xxx.xxx.xxx.xxx
```

Where you replace the last two lines with a list of hosts you want to allow to access your site. As you can see, hosts can be listed either by host and domain name, or by IP address.

You are now finished setting up Acidlab. If you make a mistake during the Acidlab

configuration utility, you can access it again later by running the command:

```
dpkg-reconfigure acidlab
```

Appendix E – Switch Reference

The following is a list of Snort's switches and what their function is:

-A mode	Allows you to override the alert logging mode if you are logging alerts to an alert file. The four available modes are: fast, which logs single-line alerts; full, which logs complete alerts; non, which disables alerting; and unsock, which sends alerts across a UNIX socket.
-b	Logs all packets to a file in binary.
-B mask	Modifies a range of IP addresses based on the specified mask so they do not appear in plain form in logs. Must be used with -h switch to identify addresses to be altered.
-c file	Use the specified configuration file.
-C	Print packets in characters and not hex.
-d	When in verbose mode or when logging packets, this switch causes Snort to also display application layer data.
-D	Run as a daemon.
-e	Includes link layer data in logs.
-F file	Loads BPF filters found in the specified file.
-g group	Allows you to specify an alternate group for Snort to run in. Commonly used so that Snort does not keep root privileges after it has completed initializing.
-h net	Specify a home network. Used in conjunction with other switches.
-i interface	Specify an interface for listening.
-I	Include the name of the interface with alerts. Useful for multiple listening interfaces.
-k mode	Identify types of packets for performing checksum. The available modes are: `all`, `noip`, `notcp`, `noudp`, `noicmp`, and `none`.
-l dir	Specify a directory for file based logs.
-L name	Used with -b to log binary data to a file with a name of your choosing. The default name uses a timestamp.
-m mask	Specify mode for creating files.
-n count	Close Snort after a specified number of packets have been processed.
-N	Generate alerts, but don't log packets.
-o	Packets go through three general stages inside Snort: Alert, Pass, and Log. The default behavior is to check for alerts, pass the data, then log anything relevant. This switch changes the order to pass, alert, log.
-O	IP address will be logged as “xxx.xxx.xxx.xxx” in ASCII dumps. If you use the switch -h to specify a home network, on IP addresses in that range will be hidden.
-p	Run Snort without promiscuous mode.
-P length	Set “snaplen” for packets to the specified length.
-q	Initialize quietly, without displaying any of the normal information.
-r file	Process the given file. The file must be in tcpdump format.
-s	Alert to the system log file.
-S var=val	Set a Snort variable to a given value. The variables can all be modified through the configuration file, but this switch is good for one-time changes.
-t root	Set Snort's root directory. Output files are relative to this directory.

-T	Perform a test on all configuration files. Since Snort often runs in the background, you can use this to make sure it will be starting up properly.
-u user	Specify a user for Snort to run as once it has finished initializing.
-U	Write timestamps in UTC.
-v	Verbose mode: a lot of data is printed to the console.
-V	Get Snort's version number.
-X	Log packets exactly as they appear starting with link layer data.
-y	Log the year along with the date.
-z	Reduces the number of alerts generated by attacks that flood Intrusion Detection Systems in order to mask an actual attack.
-?	Print command-line switch quick reference.

Appendix F – The Default Rules

There are currently 48 files included in the standard rule set, each containing one or more rules, accounting for a few thousand standard rules. We will now take a look at each category of rules in the standard set and identify its purpose.

Attack-responses	These rules deal with detecting computers that have already been compromised. For example it detects forced attempts at copying files, browsing directories or running command prompts.
Backdoor	Detects common backdoor attacks like DeepThroat and MyDoom. Note that these rules are not activated by the default configuration file.
Bad-traffic	Detects traffic that should not been seen under any normal circumstances, like traffic sent on TCP port 0.
Chat	Not for intrusion detection, and not activated by default. This is for companies in which using chat or instant messaging programs is against company policy.
Ddos	Detects distributed denial-of-service attacks.
Deleted	This file contains rules that have been deleted from the standard rule set for any reason. All rules in this file are commented, and there is no entry for this file present in the configuration file (commented or otherwise).
Dns	Detects various DNS exploits.
Dos	Detects many denial-of-service attacks.
Experimental	This file is a place-holder for new experimental rules. The file is active in the Snort configuration file, but the file itself does not contain any rules.
Exploit	Detects exploits in a wide variety of technologies ranging from Netscape to Oracle and SSH. In most cases, these are overflow attempts.
Finger	Detects attempts at gaining information about network computers.
Ftp	Detects attacks through an FTP server. There are several types of attacks, including overflows, malformed commands, and attempts at gaining access.
Icmp	One of two files that detect ICMP related attacks.
Icmp-info	Another set of rules that detects ICMP attacks. This file is not used by default since it contains several general rules that trigger frequently. The other ICMP rules file contains the more important ICMP scans.
Imap	Detects common IMAP attacks.
Info	This file is not activated by default, and in general is not for intrusion detection. It mainly deals with failed login attempts to various protocols, though it contains some other rules as well.
Local	A placeholder for rules that you write. This file is included in the configuration file's list of rules.
Misc	A repository for rules that do not fit in any other category, but these rules are activated by default.
Multimedia	Used to enforce company policy rather than for intrusion detection: these rules detect common streaming media formats such as RealPlayer. This is not activated by default.
Mysql	These rules alert on failed attempts to either gain root access to a database,

	or to gain information about it's contents. Only traffic to computers listed in the $SQL_SERVERS variable is checked.
Netbios	Detects NetBIOS exploits.
Nntp	Detects NNTP overflow attempts.
Oracle	Detects attacks against Oracle servers that you included in your list of SQL servers in the configuration file variable $SQL_SERVERS.
Other-ids	Checks for signs of other Intrusion Detection Systems running on the network.
P2p	Not activated by default, used to detect use of peer-to-peer programs such as Napster.
Policy	Miscellaneous rules for detecting breach of policy. Not activated by default.
Pop2	Detects POP2 exploits.
Pop3	Detects POP3 exploits.
Porn	Not activated by default, used for company policy.
Rpc	Detects potential threats in remote procedure calls. For example, alerts will be triggered if port mapping is attempted.
Rservices	Detects attempts to exploit remote services, such as login overflows.
Scan	Detects common port scanning techniques.
Shellcode	Searches traffic for shellcode that is common to hacking attempts. These rules are disabled by default because they perform a complete search of each packet, and are therefore very intense operations. These rules can bog down your IDS.
Smtp	Detects exploits against your SMTP servers specified in the configuration variable $SMTP_SERVERS.
Snmp	Detects SNMP exploits.
Sql	Deals with exploits of the Microsoft SQL server. These rules only check traffic to computers listed in the $SQL_SERVERS variable in your configuration file.
Telnet	Checks Telnet traffic for aberrant behavior. Uses the $TELNET_SERVERS variable for a list of known Telnet servers.
Tftp	Detects TFTP exploits.
Virus	Checks email for attachments that are potentially viruses. The single rule in this file does this by simply checking the extension of attached files against a list of extensions that could possibly be viruses, such as exe, bat and com. This rule is disabled by default, since it is not to be used as a replacement for a good anti-virus system.
Web-attacks	Detects attacks against your web servers from external sources. These rules are not active by default. Checks traffic to computers listed in $HTTP_SERVERS.
Web-cgi	Detects CGI exploits against your web servers listed in $HTTP_SERVERS from external sources.
Web-client	Detects internal users sending out bad information on web ports, and attacks against internal computers via web ports.
Web-coldfusion	Detects attacks exploiting Cold Fusion on your web servers.
Web-frontpage	Detects attacks exploiting Frontpage on your web servers.

Web-iis	Detects attacks specific to Microsoft IIS web servers on your network.
Web-misc	Detects a very wide variety of web-related attacks, both against servers and users.
Web-php	Detects PHP related attacks against your web servers.
X11	Detects X related issues.