**Working with Snort Rules:**

A rule may be used to generate an alert message, log a message, or, in terms of Snort, pass the data packet, i.e., drop it silently. The word pass here is not equivalent to the traditional meaning of pass as used in firewalls and routers. In firewalls and routers, pass and drop are opposite to each other. Snort rules are written in an easy to understand syntax. Most of the rules are written in a single line.

**TCP/IP Network Layers:**

Before you move to writing rules, let us have a brief discussion about TCP/IP layers. This is important because Snort rules are applied on different protocols in these layers. TCP/IP is a five-layer protocol. These layers interact with each other to make the communication process work. The names of these layers are:

1. The physical layer.

2. The data link layer. In some literature this is also called the network interface layer. The physical and data link layers consist of physical media, the network interface adapter, and the driver for the network interface adapter. Ethernet addresses are assigned in the data link layer.

3. The network layer, which is actually IP (Internet Protocol) layer. This layer is responsible for point-to-point data communication and data integrity. All hosts on this layer are distinguished by IP addresses. In addition to IP protocol, ICMP (Internet Control Message Protocol) is another major protocol in this layer.

4. The transport layer, which is actually TCP/UDP layer in the TCP/IP protocol. TCP (Transmission Control Protocol) is used for connection-oriented and reliable data transfer from source to destination. UDP (User Datagram Protocol), on the other hand, is used for connectionless data transfer. There is no assurance that data sent through UDP protocol will actually reach its destination. UDP is used where data loss can be tolerated.

5. The application layer consists of applications to provide user interface to the network. Examples of network applications are Telnet, Web browsers, and FTP clients. These applications usually have their own application layer protocol for data communication.

**The First Bad Rule:** Here is the first (very) bad rule. In fact, this may be the worst rule ever written, but it does a very good job of testing if Snort is working well and is able to generate alerts.

***alert ip any any -> any any (msg: "IP Packet detected";)***

***alert icmp any any -> any any (msg: "ICMP Packet found";)***

The following is a brief explanation of different words used in this rule:

• The word “alert” shows that this rule will generate an alert message when the criteria are met for a captured packet. The criteria are defined by the words that follow.

• The “ip” part shows that this rule will be applied on all IP packets.

• The first “any” is used for source IP address and shows that the rule will be applied to all packets.

• The second “any” is used for the port number. Since port numbers are irrelevant at the IP layer, the rule will be applied to all packets.

• The -> sign shows the direction of the packet.

• The last part is the rule options and contains a message that will be logged along with the alert.

• The third “any” is used for destination IP address and shows that the rule will be applied to all packets irrespective of destination IP address.

• The fourth “any” is used for destination port. Again it is irrelevant because this rule is for IP packets and port numbers are irrelevant

You can use this rule at the end of the snort.conf file the first time you install Snort. The rule will generate an alert message for every captured IP packet. It will soon fill up your disk space if you leave it there.

Example: The following rule shows that only those packets that go to a single host with IP address192.168.2.113 will generate an alert:

***alert icmp any any -> 192.168.1.113/32 any \ (msg: "Ping with TTL=100"; ttl:100;)***

**Structure of a Rule:** All Snort rules have two logical parts: rule header and rule options.



The rule header contains information about what action a rule takes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Action** | **Protocol** | **Address** | **Port** | **Direction** | **Address** | **Port** |

Rule Headers: A rule header consists of the section of the rule before starting parentheses and has many parts.

Rule actions: There are five predefined actions: ***Pass, Log, Alert, Active, Dynamic***

Address: For example, an address 192.168.2.0/24 represents C class network 192.168.2.0 with 24 bits in the network mask. A network mask with 24 bits is 255.255.255.0.

Examples for masking: A class network (8 bits) – 255.0.0.0, B class network (16 bits) – 255.255.0.0, C class network (24 bits) – 255.255.255.0

For example, if you want to generate alerts for all TCP packets with TTL=100 going to web server 192.168.1.10 at port 80 from any source, you can use the following rule:

***alert tcp any any -> 192.168.1.10/32 80 (msg: "TTL=100"; \ttl: 100;)***

***The part of the rule before the starting parenthesis is called the rule header. The part of the rule before the starting parenthesis is called the rule header.***

***alert icmp any any -> any any (msg: "Ping with TTL=100"; \ ttl: 100;) (ttl-time to live)***

***The rule action: alert, Protocol: icmp***

***Source address: any any on the left side (first any source address, second any source port)***

***Destination address: any any on the right side (first any destination address, second any destination port)***

**Rule Headers:** A rule header consists of the section of the rule before starting parentheses and has many parts.

**Rule Actions: (1) Pass -** This action tells Snort to ignore the packet. It plays an important role in speeding up Snort operation in cases where you don’t want to apply checks on certain packets. **(2) Log -** The log action is used to log a packet. Packets can be logged in different ways. **(3) Alert** - The alert action is used to send an alert message when rule conditions are true for a particular packet. An alert can be sent in multiple ways.

**(4) Dynamic -** Dynamic action rules are invoked by other rules using the “activate” action. In normal circumstances, they are not applied on a packet. A dynamic rule can be activated only by an “activate” action defined in another role.

**(5)** User Defined Actions - These new action types are defined in the configuration file snort.conf. A new action is defined in the following general structure:

ruletype smb\_db\_alert

{

type alert

output alert\_smb: workstation.list

output database: log, mysql, user=rr password=rr \ dbname=snort host=localhost

}

**Protocols:** Protocol is the second part of a Snort rule. The protocol part of a Snort rule shows on which type of packet the rule will be applied. Currently Snort understands the following protocols:

• IP

• ICM

• TCP

• UDP

**Address:** There are two address parts in a Snort rule. These addresses are used to check the source from which the packet originated and the destination of the packet. The address may be a single IP address or a network address.

For example, an address 192.168.2.0/24 represents C class network 192.168.2.0 with 24 bits in the network mask. A network mask with 24 bits is 255.255.255.0. Keep the following in mind about number of bits in the netmask:

***• If the netmask consists of 24 bits, it is a C class network.***

***• If the netmask consists of 16 bits, it is a B class network.***

***• If the netmask consists of 8 bits, it is an A class network.***

***• For a single host, use 32 bits in the netmask field.***

**Address Exclusion:** Snort provides a mechanism to exclude addresses by the use of the negation symbol **!**, an exclamation point. This symbol is used with the address to direct Snort not to test packets coming from or going to that address. For example, the following rule is applied to all packets except those that originate from class C network 192.168.2.0.

***alert icmp ![192.168.2.0/24] any -> any any \ (msg: "Ping with TTL=100"; ttl: 100;)***

**Port Number:** The port number is used to apply a rule on packets that originate from or go to a particular port or a range of ports.

The following rule is applied to all packets that originate from a Telnet server in 192.168.2.0/24, which is a class C network and contains the word “confidential”:

***alert tcp 192.168.2.0/24 23 -> any any \ (content: "confidential"; msg: "Detected confidential";)***

**Port Ranges:** You can also use a range of ports instead of only one port in the port field. Use a colon to separate starting and ending port numbers. For example, the following rule will create an alert for all UDP traffic coming from ports 1024 to 2048 from all hosts.

***alert udp any 1024:2048 -> any any (msg: “UDP ports”;)***

**Rule Options:** Rule options follow the rule header and are enclosed inside a pair of parentheses. There may be one option or many and the options are separated with a semicolon. If you use multiple options, these options form a logical AND. The action in the rule header is invoked only when all criteria in the options are true.

**The ack Keyword:** The TCP header contains an Acknowledgement Number field which is 32 bits long. The field shows the next sequence number the sender of the TCP packet is expecting to receive.

***alert tcp any any -> 192.168.1.0/24 any (flags: A; \ ack: 0; msg: "TCP ping detected";)***

**The classtype Keyword:** Rules can be assigned classifications and priority numbers to group and distinguish them. To fully understand the classtype keyword, first look at the file classification.config which is included in the snort.conf file using the include keyword.

Each line in the classification.config file has the following syntax:

***config classification: name,description,priority***

***Example: config classification: DoS,Denial of Service Attack,2***

***alert udp any any -> 192.168.1.0/24 6838 (msg:"DoS"; \ content: "server"; classtype:DoS;)***

***alert udp any any -> 192.168.1.0/24 6838 (msg:"DoS"; \ content: "server"; classtype:DoS; priority:1)***

**The content Keyword:** One important feature of Snort is its ability to find a data pattern inside a packet. The pattern may be presented in the form of an ASCII string or as binary data in the form of hexadecimal characters.

The following rule detects a pattern “GET” in the data part of all TCP packets that are leaving 192.168.1.0 network and going to an address that is not part of that network. The GET keyword is used in many HTTP related attacks; however, this rule is only using it to help you understand how the content keyword works.

***alert tcp 192.168.1.0/24 any -> ![192.168.1.0/24] any \ (content: "GET"; msg: "GET matched";)***

The following rule does the same thing but the pattern is listed in hexadecimal.

***alert tcp 192.168.1.0/24 any -> ![192.168.1.0/24] any \ (content: "|47 45 54|"; msg: "GET matched";)***

There are three other keywords that are used with the content keyword. These keywords add additional criteria while finding a pattern inside a packet. These are:

**• The offset keyword:** Using this keyword, you can start your search at a certain offset from the start of the data part of the packet. Use a number as argument to this keyword.

The following rule starts searching for the word “HTTP” after 4 bytes from the start of the data.

***alert tcp 192.168.1.0/24 any -> any any \ (content: "HTTP"; offset: 4; msg: "HTTP matched";)***

**• The depth keyword:** The depth keyword is also used in combination with the content keyword to specify an upper limit to the pattern matching.

The following rule tries to find the word “HTTP” between characters 4 and 40 of the data part of the TCP packet:

***alert tcp 192.168.1.0/24 any -> any any (content: \ "HTTP"; offset: 4; depth: 40; msg: "HTTP matched";)***

**• The nocase keyword:** The nocase key word is used to make the search case-insensitive.

**The dsize Keyword:** The dsize keyword is used to find the length of the data part of a packet. Many attacks use buffer overflow vulnerabilities by sending large size packets. Using this keyword, you can find out if a packet contains data of a length larger than, smaller than, or equal to a number.

The following rule generates an alert if the data size of an IP packet is larger than 6000 bytes.

***alert ip any any -> 192.168.1.0/24 any (dsize: > 6000; \ msg: "Large size IP packet detected";)***

**The react Keyword:** The react keyword is used with a rule to terminate a session to block some sites or services.

***alert tcp 192.168.1.0/24 any -> any 80 (msg: "Outgoing \ HTTP connection"; react: block;)***

**The reference Keyword:** The reference keyword can add a reference to information present on other systems available on the Internet. It does not play any role in the detection mechanism itself and you can safely ignore it as far as writing Snort rules is concerned.

***alert udp $EXTERNAL\_NET any -> $HOME\_NET 1900 \ (msg:"MISC UPNP malformed advertisement"; \ content:"NOTIFY \* "; nocase; classtype:misc-attack; \ reference:cve,CAN-2001-0876; reference:cve, \ CAN-2001-0877; sid:1384; rev:2;)***

**The rev Keyword:** The rev keyword is added to Snort rule options to show a revision number for the rule. If you are updating rules, you can use this keyword to distinguish among different revision. Output modules can also use this number to identify the revision number.

The following rule shows that the revision number is 2 for this rule:

***alert ip any any -> any any (ipopts: lsrr; \ msg: "Loose source routing attempt"; rev: 2;)***

***(Options for ipopts are Record Route (rr), Time Stamps (ts), Loose source routing (lsrr), Strict source routing (ssrr)).***

**The Snort Configuration File:**

Snort uses a configuration file at startup time. A sample configuration file snort.conf is included in the Snort distribution. Snort uses a configuration file at startup time. A sample configuration file snort.conf is included in the Snort distribution.

***/opt/snort/snort -c /opt/snort/snort.conf***

This file contains six basic sections:

(1) Variable definitions, where you define different variables.

(2) Config parameters. These parameters specify different Snort configuration options.

(3) Preprocessor configuration. Preprocessors are used to perform certain actions before a packet is operated by the main Snort detection engine.

(4) Output module configuration. Output modules control how Snort data will be logged.

(5) Output module configuration. Output modules control how Snort data will be logged.

(6) Rules configuration and include files. Although you can add any rules in the main snort.conf file, the convention is to use separate files for rules. These files are then included inside the main configuration file using the include keyword.

**Using Variables in Rules:** In the configuration file, you can use variables. This is a very convenient way of creating rules. For example, you can define a variable HOME\_NET in the configuration file.

***var HOME\_NET 192.168.1.0/24***

Later on you can use this variable HOME\_NET in your rules:

***alert ip any any -> $HOME\_NET any (ipopts: lsrr; \ msg: “Loose source routing attempt”; sid: 1000001;)***

**Preprocessor Configuration:** Preprocessors or input plug-ins operate on received packets before Snort rules are applied to them. The preprocessor configuration is the second major part of the configuration file.

The general format of configuring a preprocessor is as follows:

***Preprocessor <preprocessor-name> [:<configuration-options>]***

The following is an example of a line in the configuration file for IP defragmentation preprocessor frag2: ***preprocessor frag2***

The following is an example of a stream4 preprocessor with an argument to detect port scans.

***preprocessor stream4: detect\_scans***

**Output Module Configuration:** Output modules, also called output plug-ins, manipulate output from Snort rules. For example, if you want to log information to a database or send SNMP traps, you need output modules.

The following is the general format for specifying an output module in the configuration file:

***Output <output-module-name> [:<configuration-options>]***

For example, if you want to store log messages to a MySQL database, you can configure an output module that contains the database name, database server address, user name and password.

***output database: alert, mysql, user=rr password=boota \ dbname=snort host=localhost***

**Order of Rules Based upon Action:** The five types of the rules can be categorized into three basic types.

1. Alert rules

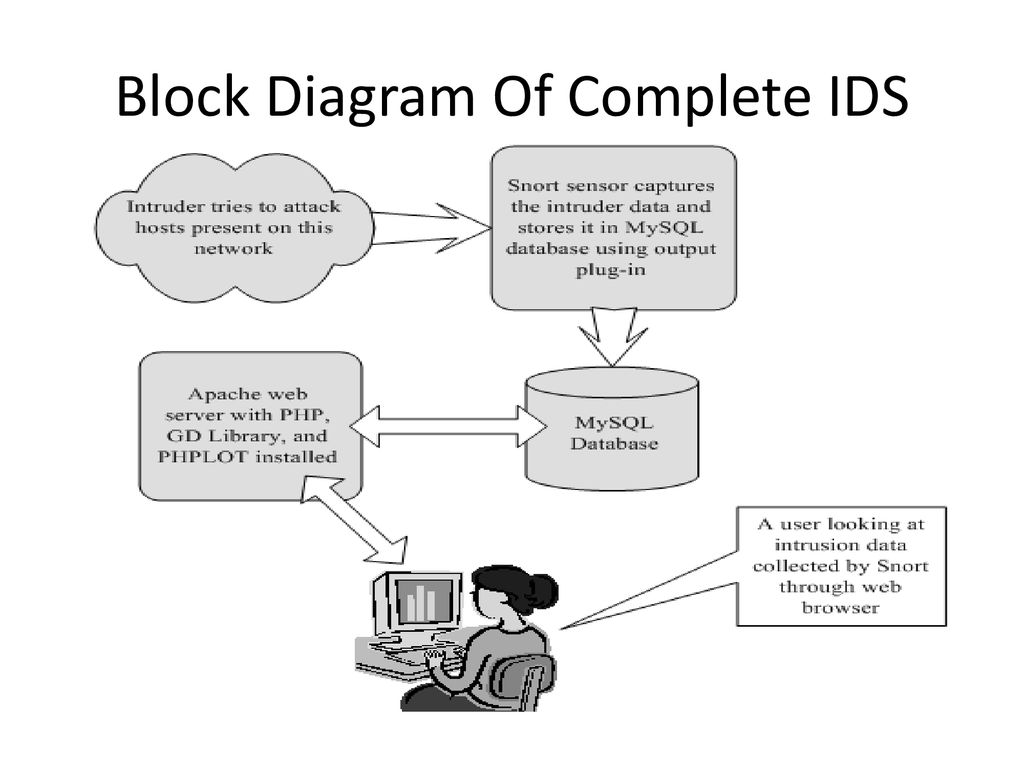
2. Pass rules

3. Log rule

**Plugins, Preprocessors and Output Modules:**

Preprocessors and output modules are two important parts of Snort architecture. Preprocessors process received data packets before rules are applied to them. Output modules control output generated from Snort’s detection mechanism.

Snort allows you to select which preprocessors and output modules should be enabled. From a user standpoint, this is done through the Snort configuration file snort.conf. Preprocessors and Output modules are also called plug-ins in some literature.



**Preprocessors:** When a packet is received by Snort, it may not be ready for processing by the main Snort detection engine and application of Snort rules. For example, a packet may be fragmented. Before you can search a string within the packet or determine its exact size, you need to defragment it by assembling all fragments of the data packet. The job of a preprocessor is to make a packet suitable for the detection engine to apply different rules to it.

During the installation process, you can compile support of different preprocessors into Snort. Configuration parameters for different preprocessors (also called input plug-ins and input modules) are present in the snort.conf file. Using the file, you can enable or disable different preprocessors.

All preprocessors are enabled in the Snort configuration file using the preprocessor keyword. The general format of enabling a preprocessor is as follows: ***preprocessor frag2***

**HTTP Decode:** The Hyper Text Transfer Protocol (HTTP) allows intrusion detection systems to use hexadecimal characters in URI to defeat known attacks. For example, this can be done by inserting something like %3A%2F%2F in the URI to replace :// characters.

The following line in the configuration will apply HTTP decode for packets coming to ports 80, 8080, 443.

***preprocessor http\_decode: 80 8080 443***

**Port Scanning:** Port scanning is a process of finding out which ports are open on a particular host or all hosts on a network. The first step in any intruder activity is usually to find out what services are running on a network. Once an intruder has found this information, attacks for known vulnerabilities for these services are tried. The portscan preprocessor is designed to detect port scanning activities. The preprocessor can be used to log the port scanning activities to a particular location in addition to standard logging. Hackers can use multiple port scanning methods.

The following is the general format of the preprocessor used in the snort.conf file.

***preprocessor portscan:<address><ports><time period><file>***

The following line in the Snort configuration file is used to detect port scanning on network 192.168.1.0/24 and to log activity in /var/log/snort/portscan.log file.

***preprocessor portscan: 192.168.1.0/24 5 10 \ /var/log/snort/portscan.log***

You can also use another preprocessor in conjunction with this preprocessor. This preprocessor is portscan-ignorehosts, which can be used to ignore some hosts if any port scanning activity is detected from them. The following line in the configuration file will ignore two hosts, 192.168.1.10 and 192.168.1.13.

***preprocessor portscan-ignorehosts: 192.168.1.10/32 \ 192.168.1.13/32***

**The frag2 Module:** This preprocessor does IP packet defragmentation. Old versions of Snort used another preprocessor named defrag. The frag2 preprocessor uses a splay tree algorithm, which is a self-organizing data structure. For configuration, use and administration of Snort, you need not understand this algorithm. ***preprocessor frag2***

The following command configures the preprocessor with 2MB memory and a timeout period of 30 seconds: ***preprocessor frag2: 2097152, 30***

**The stream4 Module:** Stream4 is a replacement for the Stream module used in older versions of Snort. It provides two basic functions:

1. TCP stream reassembly

2. Stateful inspection

You must configure two preprocessors in the snort.conf file for Stream4 to work properly. These modules are “stream4” and “stream4\_reassemble.” Both of these take a number of arguments. If you don’t specify an argument, a default value is used instead.

**The general format of stream4 preprocessor is as follows:**

***preprocessor stream4: [noinspect], [keepstats], \ [timeout], [memcap], [detect\_scan], \ [detect\_state]***

**The general format of the stream4\_reassemble preprocessor is as follows:**

***preprocessor stream4\_reassemble: [clientonly], [serveronly],[noalerts],[ports<ports-list]***

**ARP Spoofing:** Address Resolution Protocol (ARP) is used to find a MAC address when an IP address is known. ARP is needed when a host wants to send an IP packet to another host on the local network. The sending host broadcasts an ARP packet on the network asking, “Who has this IP address?” The host who has that IP address will respond with its MAC address. After that, the sending host will send the data packet (usually called a frame at the link layer level) to the destination host.

The following entry in the Snort configuration file (snort.conf) will configure this preprocessor and will detect unicast anomalies:

***preprocessor arpspoof: -unicast***

The following line adds an IP address and MAC address pair which can be used later on to detect ARP spoofing attempts.

***preprocessor arpspoof\_detect\_host: 192.168.1.13 \ 34:45:fd:3e:a2:01***

**Output Modules:** Output modules are used to control the output from Snort detection engine. By default, the output from alerts and logs go into files in the /var/log/snort directory. Using output modules, you can process output and send output messages a number of other destinations. Commonly used output modules are:

• The database module is used to store Snort output data in databases.

• The SNMP module can be used to send Snort alerts in the form of traps to a management server.

• The SMB alerts module can send alerts to Microsoft Windows machines in the form of pop-up SMB alert windows.

• The syslog module logs messages to the syslog utility. Using this module, you can log messages to a centralized logging server.

• You can also use XML or CSV modules to save data in XML or comma separated files. The CSV files can then be imported into databases or spreadsheet software for further processing or analysis.

For example, if you want to log messages to MySQL database called “snort” using database user name “rr” and password “rr” located on the same machine where Snort is running, you use the following line in snort.conf file.

***output database: log, mysql, user=rr password=rr \ dbname=snort host=localhost***

Sometimes you may want to send alerts to multiple locations. Defining your own action using the ruletype keyword is a good idea. For example, the following lines in the snort.conf file will define an action type called “smb\_db\_alert” that will cause alerts to be sent to both the database and SMB pop-up windows for rules that use this action type:

**ruletype smb\_db\_alert**

**{**

**type alert**

**output alert\_smb: workstation.list**

**output database: log, mysql, user=rr password=rr \ dbname=snort host=localhost**

**}**

The following rule uses this new action type. Alerts generated by this rule will go to MySQL database as well as to the Windows machine in the form of pop-up windows:

***smb\_db\_alert icmp any any -> 192.168.1.0/24 any \ (fragbits: D; msg: "Dont Fragment bit set";)***

**The alert\_syslog Output Module:** Syslog is a system logging daemon available on almost all UNIX systems. It uses a configuration file /etc/syslog.conf where you can define different parameters to determine what happens when a message for a defined facility is received.

The alert\_syslog module allows you to send alerts to the syslog facility. The Syslog daemon can also be used to forward alerts to some other host as well if you need centralized logging. The following is the general format for using this module.

***output alert\_syslog: <facility><priority><options>***

**Facility names that can be used with this module are:**

• LOG\_AUTH • LOG\_AUTHPRIV • LOG\_DAEMON • LOG\_LOCAL0 • LOG\_LOCAL1 • LOG\_LOCAL2 • LOG\_LOCAL3 • LOG\_LOCAL4 • LOG\_LOCAL5 • LOG\_LOCAL6 • LOG\_LOCAL7 • LOG\_USER

**Priorities that are available with this module are:**

• LOG\_EMERG • LOG\_ALERT • LOG\_CRIT • LOG\_ERR • LOG\_WARNING • LOG\_NOTICE • LOG\_INFO • LOG\_DEBUG

***Note that LOG\_EMERG is the highest priority and LOG\_DEBUG is the lowest priority.***

**Options that you can use with this module are:**

• LOG\_CONS • LOG\_NDELAY • LOG\_PERROR • LOG\_PID

**Using Snort with MySQL:**

**­­** All systems need some type of efficient logging feature, usually using a database at the backend. Snort can be made to work with MySQL, Oracle or any other Open Database Connectivity (ODBC) compliant databases.

**Making Snort Work with MySQL:**

There are a few basic steps to make Snort work with MySQL. A high level step-by-step approach to build a Snort-MySQL system follows.

1. Compile Snort with MySQL support and install it. Make sure that Snort is working properly by creating some alert messages.

2. Install MySQL and use mysql client to make sure the database is available. See Appendix C for basic information about how to get started with MySQL.

3. Create a database on the MySQL server for Snort. I have named this database “snort.” You may choose any name for the database. This is explained later in this chapter.

4. Create a user name and password in the database. The user name will be used by Snort to log data.

5. Create tables in this database using scripts that came with Snort distribution in the contrib directory.

6. Modify the snort.conf file to enable the database plug-in as explained later. You will use the database name, user name and password for the database that you just created.

7. Restart Snort. If everything goes well, Snort will start logging to the database.

8. Generate some alerts and use the mysql client program to make sure that alerts are being logged into the database.

**Step 1: Snort Compilations with MySQL Support**

Snort must be compiled with --with-mysql if you want to use MySQL database with Snort. This is done with the help of the configure script as explained in Chapter 2. A typical configure script command line follows:

***./configure --prefix=/opt/snort --with-mysql=/usr/lib/mysql***

**Step 2: Install MySQL**

MySQL database server and client software in the source code form from its web site at http://www.mysql.org and compile and install it yourself.

**Step 3: Creating Snort Database in MySQL**

Once you have compiled Snort with MySQL support, the next step is to create MySQL database where Snort can log data. Before you start using MySQL, make sure that MySQL server is running on the machine that is being used as the database server.

Assuming MySQL server is running on localhost, a typical mysql session to create the database and check its status is as follows:

**[root@laptop]# mysql -h localhost -u root -p**

**Enter password:**

**Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 40 to server version: 3.23.36**

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

mysql> create database snort:

mysql> use snort

mysql> status

**Step 4: Creating MySQL User and Granting Permissions to User and Setting Password**

Using the database user root to access the Snort database is not recommended. For this purpose, you will create a new user “rr”. The next command creates a user with name rr. The same command also grants the following permissions to all tables in the snort database we recently created.

• CREATE, used to create new objects

• INSERT, used to insert data into the database

• DELETE, used to delete data from the database

• UPDATE, used to modify records

• SELECT, used to display and select records

We shall use this user to access the Snort database. This user name and password are also used in the snort.conf file when you configure output database module:

**mysql> grant CREATE, INSERT, DELETE, UPDATE, SELECT on snort.\* to rr@localhost;**

The permission for this newly created user is granted only for the database Snort. A single command creates the user and grants permission.

Now you need to assign a password to this user. The following command assigns a password “rr78x” to this user.

**mysql> set password for rr = password('rr78x');**

**Step 5: Creating Tables in the Snort Database**

**[root@laptop]# mysql -h localhost -u rr -p snort < contrib/ create\_mysql**

**Enter password:**

To display what tables have been created, use the following session:

**[root@laptop]# mysql -h localhost -u rr -p snort**

**Enter password:**

**Step 6: Modify snort.conf Configuration File**

After configuring the database and creating tables and user, you need to edit the snort.conf file. These lines in the file will enable logging of log messages to the MySQL database:

***output database: log, mysql, user=rr password=rr78x \ dbname=snort host=localhost***

The database is located on MySQL server running on the localhost, the machine where Snort is installed. If you have a separate database server, you can specify the name of the server on this line in the snort.conf file. For example if the database server is not the same as where Snort is running, you can use the following lines in the snort.conf file.

***output database: log, mysql, user=rr password=rr78x \ dbname=snort host=192.168.1.23***

**Step 7: Starting Snort with Database Support**

When you start Snort after database configuration, the starting message shows what database is being used. The boldface lines show database related information.

***[root@laptop]# /opt/snort/bin/snort -c /etc/snort/snort.conf Log directory = /var/log/snort***

**Step 8: Logging to Database**

After configuring the database properly, you should check if log and alert messages are being saved in the database tables. We use the following two rules for Snort to test the database.

***alert ip any any -> any any (ipopts: lsrr; msg: \ "LSRR Options set"; logto: "test";)***

***alert icmp any any -> 192.168.1.0/24 any (fragbits: D; \ msg: "Dont Fragment bit set";)***

**Secure Logging to Remote Databases Securely Using Stunnel:**

The MySQL database server is listening to port number 3306. If your database server is not on the same machine where Snort is running, you have to log messages on a remote database server. From a security point of view, you may want to encrypt traffic between Snort and the database server. Stunnel or Secure Tunnel is an open source package available from http://www.stunnel.org that provides you a secure tunnel between two hosts.

***stunnel -P/tmp/ -p stunnel.pem -d 3307 -r localhost:3306***

On the Snort machine, use the following command:

***stunnel -P/tmp/ -c -d 3306 -r SERVER\_NAME:3307***

**Snort Database Maintenance:**

From time to time, you need to perform some operations on the database to keep it running efficiently. Table optimization enhances the database efficiency. You can optimize individual tables using the optimize command.

**mysql> optimize table data;**