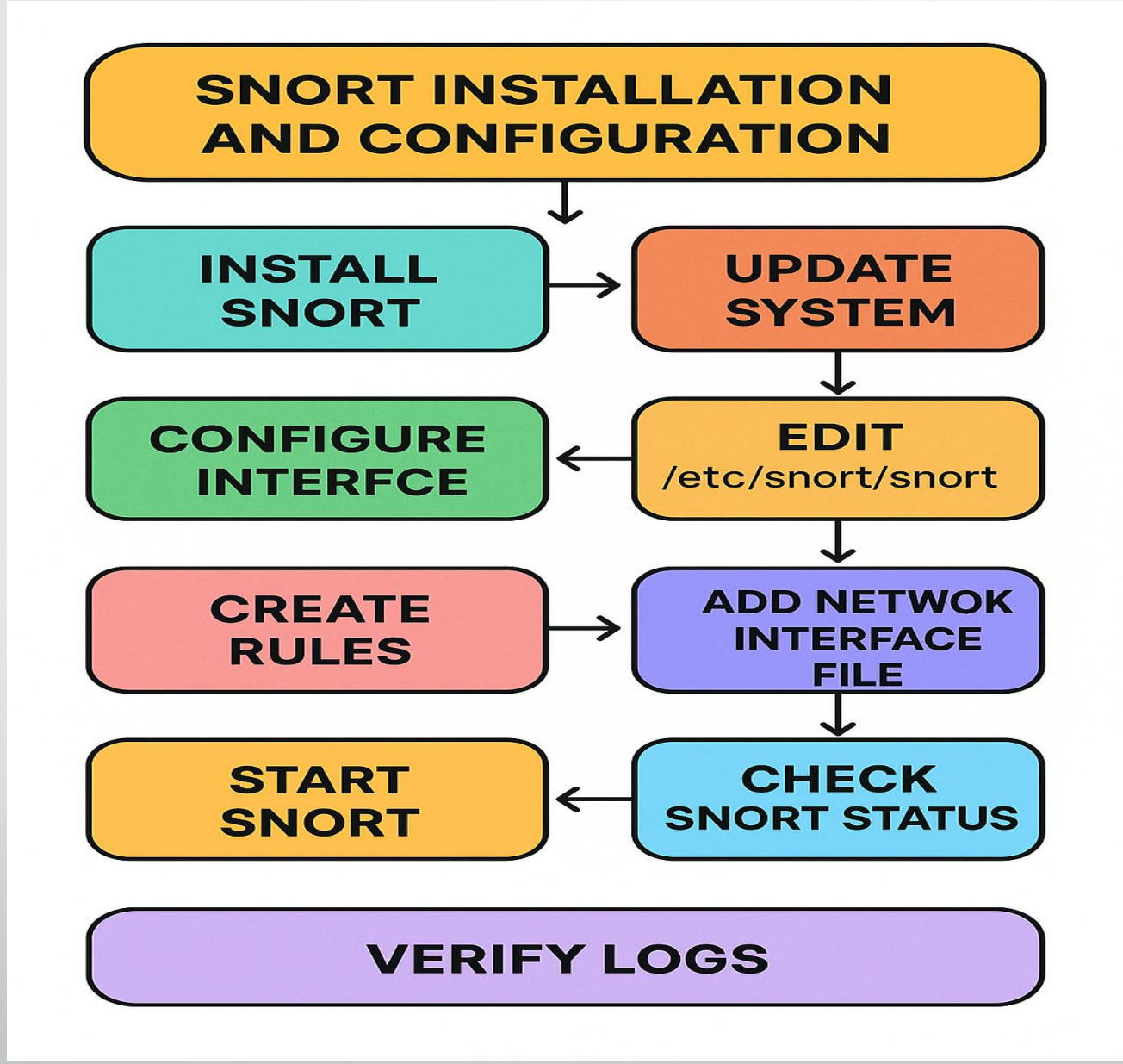
Unit III - SNORT

Introduction to Snort, Snort Installation Scenarios, Installing Snort, Running Snort on Multiple Network Interfaces, Snort Command Line Options. Step-By-Step Procedure to Compile and Install Snort Location of Snort Files, Snort Modes Snort Alert Modes- Working with Snort Rules, Rule Headers, Rule Options, The Snort Configuration File etc. Plugins, Preprocessors and Output Modules, Using Snort with MySQL

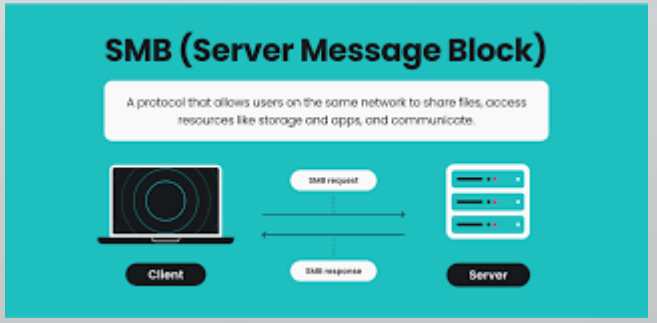
Introduction to Snort

Snort Installation Scenarios

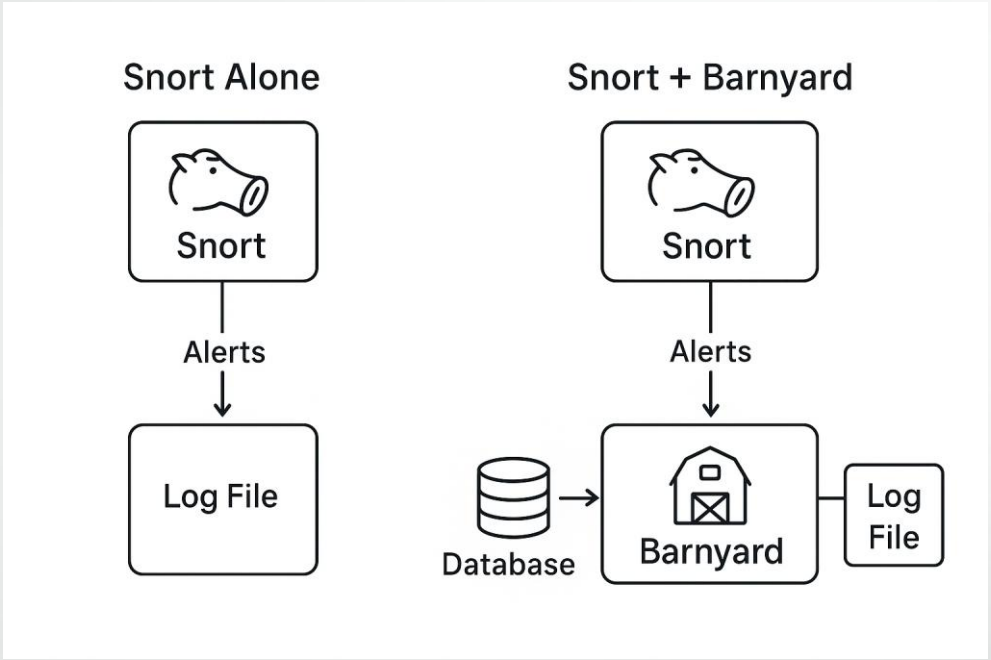

SNORT INSTALLATION AND CONFIGURATION
INSTALL SNORT
UPDATE SYSTEM
CONFIGURE INTERFCE
EDIT /etc/snort/snort
CREATE RULES
ADD NETWOK INTERFACE FILE
START SNORT
CHECK SNORT STATUS
VERIFY LOGS

SNORT INSTALLATION

- **Basic Installation:** Only the Snort daemon.
  - Captures intrusion data (text or binary files).
  - View later using a text editor or tools like **Barnyard**.
  - Sends alerts to:
    - **SNMP manager** (e.g., HP OpenView, OpenNMS(open-source network management system)).
    - **Windows machine** via SMB(Server Message Block) pop-up.



SNORT vs BARNYARD



Snort generates alert data in a **raw or unified format** (not always easy to read).
Barnyard **reads this raw data** and converts it into a more usable format.
It then **stores the data in a database** (like MySQL or PostgreSQL) or sends it to other tools for analysis.

SNORT INSTALLATION

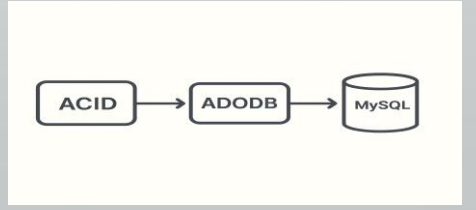
- **Complete Installation:** Snort + additional tools.
  - Logs data to a database.
  - Provides a **web interface** for analyzing and viewing alerts.
  - Makes it easier to interpret intrusion data without going through raw log files.

SNORT

- **MySQL is used with Snort** to log alert data. Other databases like Oracle can also be used but MySQL is the most popular database with
- Snort. In fact, **any ODBC (Open Database Connectivity)-compliant database** can be used with Snort.
- Apache acts as a web server.
- PHP is used as an interface between the web server and MySQL database.
- **ACID(Analysis Console for Intrusion Databases)** is a PHP package that is used to view and analyze Snort data using a web browser.
- GD library is used by ACID to create graphs.
- PHPLOT is used to present data in graphic format on the web pages

used in ACID. GD library must be working correctly to use PHPLOT.

- **ADODB(Active Data Objects DataBase)** is used by ACID to connect to MySQL database- ACID uses ADODB to **send queries** to the database (like "get all alerts from today") and to **receive results**.



SNORT INSTALLATION SCENARIOS

- Snort installations **vary depending on the network size, purpose, and environment.**
- **Test Installation**
- **Purpose:** Used for testing Snort in a simple setup.
- **Setup:**
  - Single Snort sensor-A **Snort sensor** is the part of your network security setup that **actively watches network traffic, checks it against rules, and alerts you to possible threats**..
  - Logs data in **text files**.
  - Data viewed manually by the administrator.
- **Limitations:**
  - High analysis cost in production.
  - Not suitable for large-scale environments.
- **Installation:**
  - Pre-compiled versions available at www.snort.org.
  - Linux: RPM package.
  - Windows: Executable files.

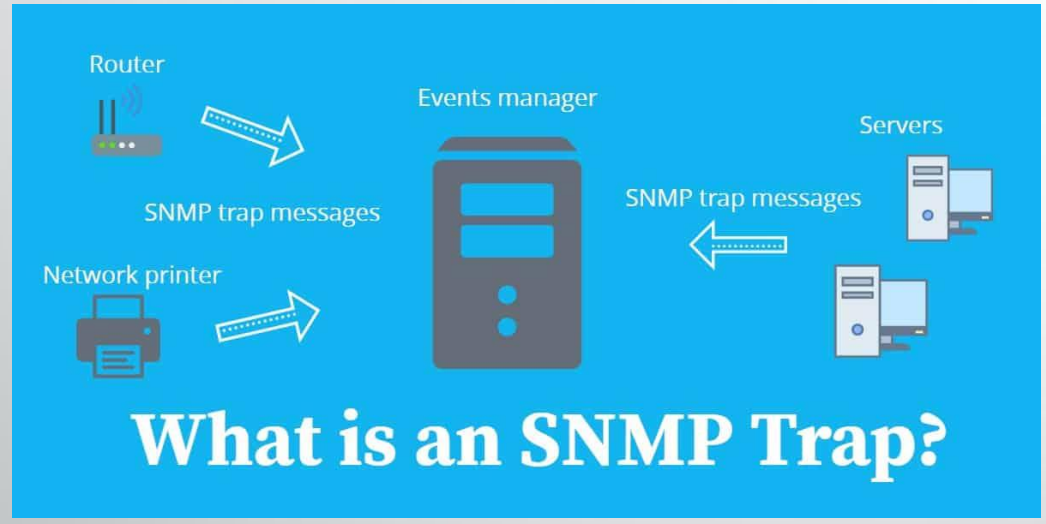
SNORT INSTALLATION SCENARIOS

- **Single Sensor Production IDS**
- **Purpose:** For small networks with one Internet connection.
- **Sensor Placement:**
  - **Behind firewall** – Detect internal intrusions.
  - **Outside firewall** – Monitor all Internet traffic.
- **Installation:**
  - Download pre-compiled or compile from source if extra features are needed.
  - Startup scripts ensure Snort runs automatically at boot.
- **Logging:**
  - Data stored in text/binary files.
  - Tools like **SnortSnarf** used for analysis(**SnortSnarf can convert logs into readable reports for easier understanding of attacks**.).

SNORT INSTALLATION SCENARIOS

- **Single Sensor with Network Management Integration**
- **Purpose:** Integrates Snort with enterprise network management systems.
- **Supported Systems:** Hewlett-Packard, IBM, Computer Associates.
- **Integration:**
  - Uses **SNMP traps** to send alerts.
  - Requires SNMP capability during Snort compilation.

SNMP Trap Workflow



- *The **Simple Network Management Protocol (SNMP)** is a widely used network protocol that facilitates the management and monitoring of devices on IP networks, such as routers, switches, servers, and printers. It enables network administrators to collect, organize, and monitor information about these devices. The service ensures optimal performance and the detection of potential issues.*

A **Trap** is an *unsolicited* message sent by an SNMP-enabled device (agent) to the Network Management System (NMS) when an event occurs.
Unlike regular SNMP polling (where the NMS asks for info), **traps are pushed automatically** to the manager.

SNMP Trap Workflow

- **SNMP Agent** runs on a device (e.g., router).
- A critical event happens (e.g., interface down, high CPU, link failure).
- The agent **sends a trap message** to the **SNMP Manager (NMS)**.
- The NMS logs, displays, or triggers alerts/actions.

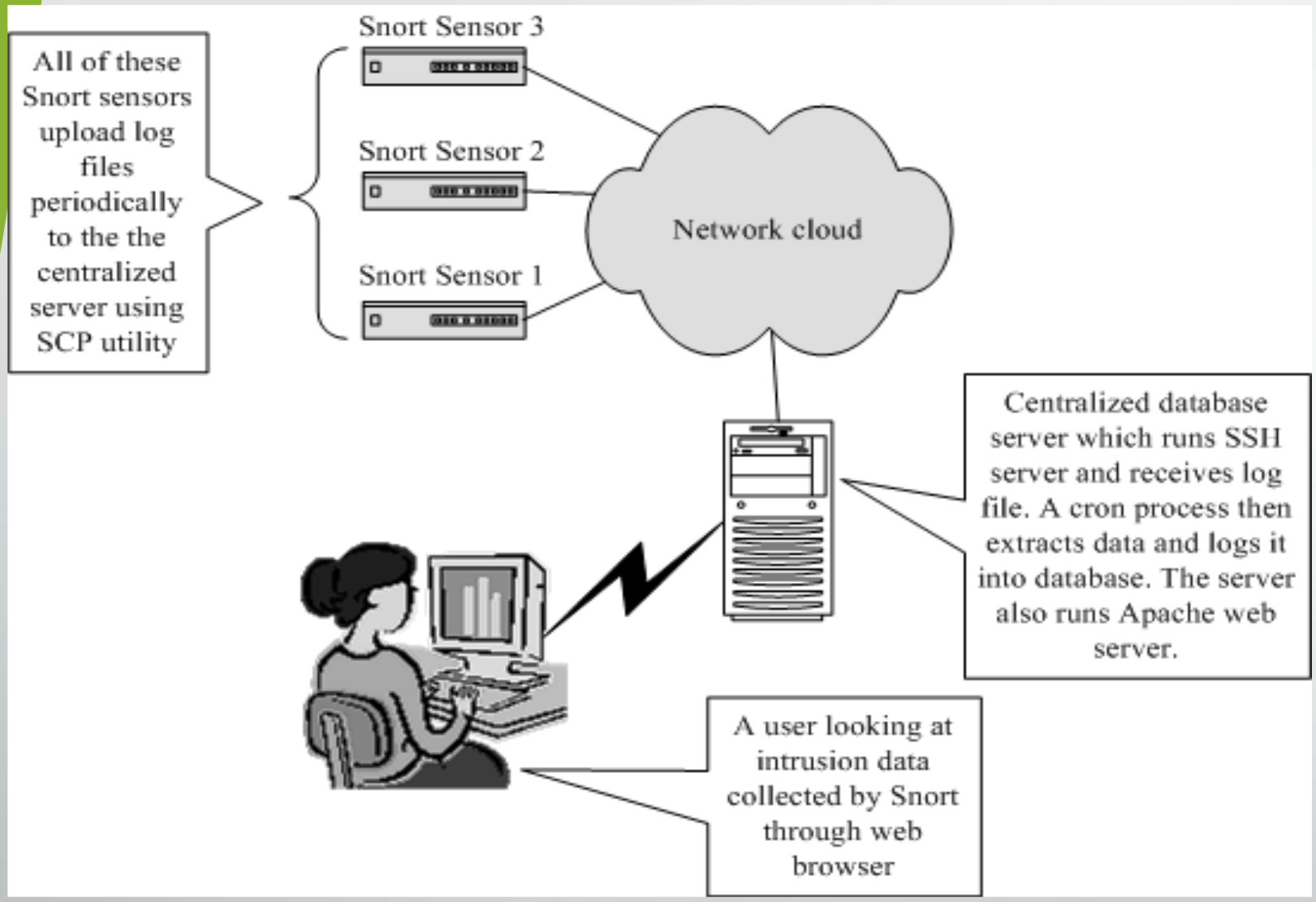
SNORT INSTALLATION SCENARIOS

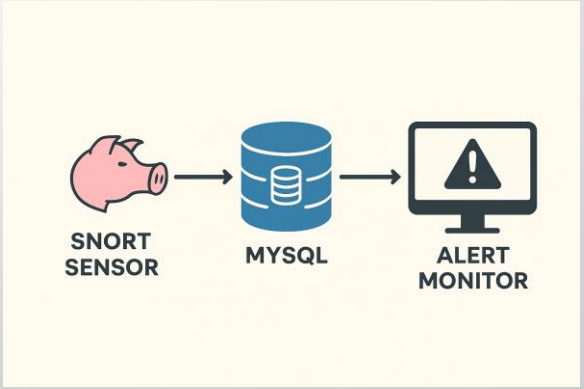
- **Multiple Sensors with Centralized Database**
- **Purpose:** Suitable for large or corporate networks with multiple locations.
- **Working:**
  - Multiple Snort sensors connect to one centralized database.
  - Data analyzed through a web server (e.g., Apache).
- **Challenges:**
  - Sensors must connect to the database when Snort starts.
  - Database must be available at all times.
  - Additional firewall ports may be needed.
- **Alternate Approach:**
  - Sensors log data locally.
  - Data uploaded periodically using **SCP (Secure Copy)** over SSH (port 22).
  - Data is slightly delayed, not real-time.

Secure Copy Protocol (SCP), a network protocol for securely transferring files using SSH; and the SCP Foundation's "Secure, Contain, Protect" acronym- anomalous objects (SCP objects) it studies and contains.

Multiple Snort Sensors Placement

Network Zone	Sensor Placement	Purpose
Internet Gateway	Outside firewall	Monitor all external traffic
Corporate LAN	Behind firewall	Monitor internal traffic for insider threats
DMZ (web servers)	In DMZ	Monitor attacks on public servers
Database Server Zone	Near critical databases	Detect unauthorized access





Distributed Snort installation with the help of tools like SCP and Barnyard.

Running Snort on Multiple Network Interfaces

Running Snort on Multiple Network Interfaces

- When you start Snort, it listens to traffic on one interface.
- Using the command line option –i <interface_name> , you can specify the interface on which you want to run it.
- **If you want to listen to multiple network interfaces, you have to run multiple copies of Snort in parallel.**
- As an example, the following two commands start listening to network interfaces etho and eth1 on a Linux machine.

Step 1: Prepare separate log directories (optional but recommended)

- mkdir -p /var/log/snort/etho
- mkdir -p /var/log/snort/eth1

Step 2: Run Snort on the first interface (etho)

- sudo snort -i etho -c /etc/snort/snort.conf -l /var/log/snort/etho -D
- -i etho → Listen on interface etho.
- -c /etc/snort/snort.conf → Load configuration and rules.
- -l /var/log/snort/etho → Store logs for this interface.
- -D → Run in background (daemon mode).

Step 3: Run Snort on the second interface (eth1)

- sudo snort -i eth1 -c /etc/snort/snort.conf -l /var/log/snort/eth1 -D
- Same as above but for eth1.

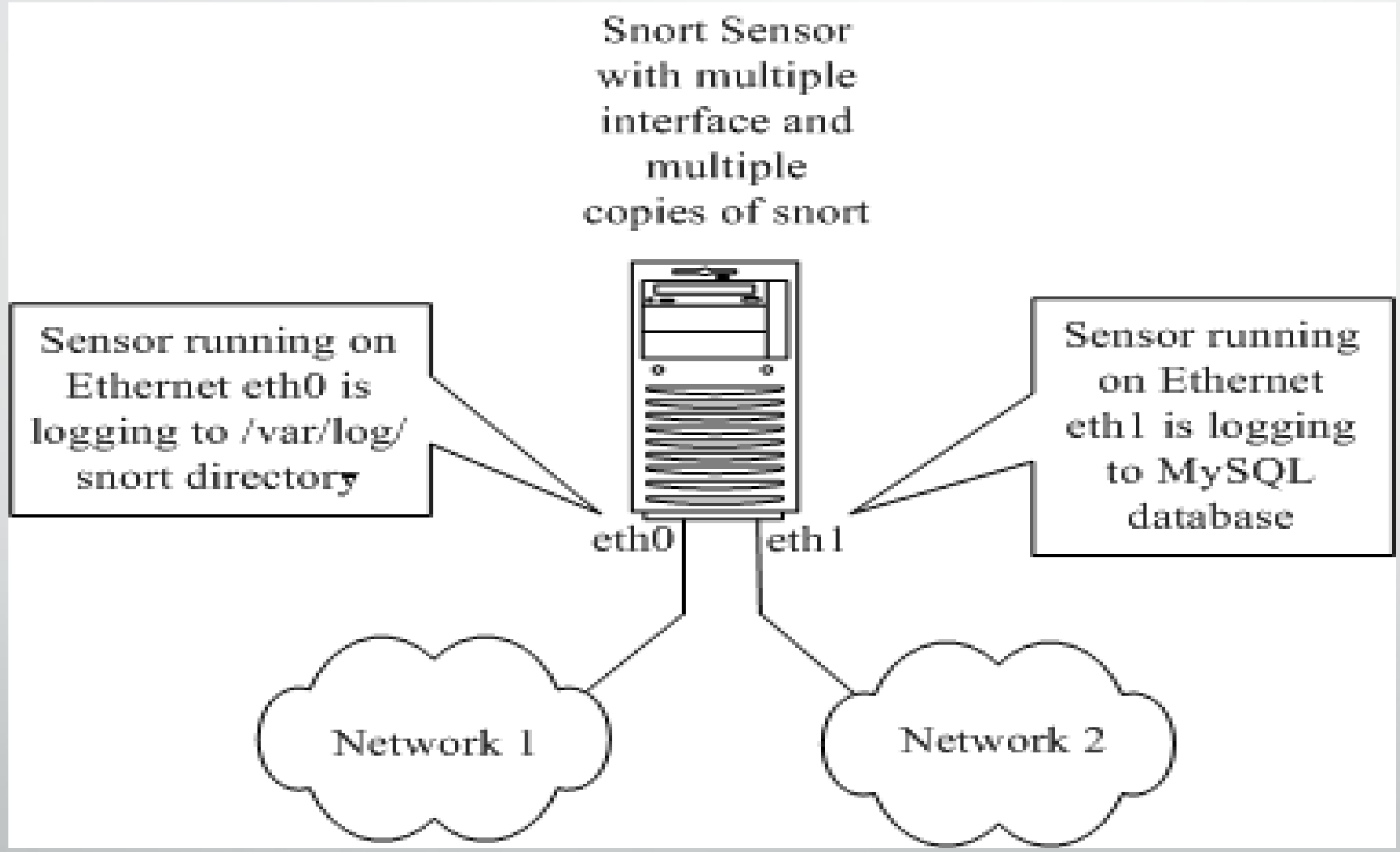
Step 4: Verify both processes are running

- ps -ef | grep snort

Step 5: Stop Snort (if needed)

- sudo killall snort

Running Snort on Multiple Network Interfaces



Snort Command Line Options.

Snort Command Line Options

- Snort has many command line options that are very useful for starting Snort in different situations.
- Basic Snort Command Format
- snort [options]

Alert Modes (-A)

-A fast (default)

- Writes alerts in a single-line format (fast and simple).
- Good for quick reviews.
- **Example: snort -A fast -c /etc/snort/snort.conf -i etho**

-A fast → Alerts will be written in a simple one-line format (fast, less detailed).

-c /etc/snort/snort.conf → Specifies the Snort configuration file (contains rules, preprocessors, etc.).

-i etho → Monitors the etho network interface

Adv:

Quick overview of alerts without too much detail.

Suitable for live monitoring with less storage.

Alert Modes (-A)

Full Alert Mode

- **Example: snort -A full -c /etc/snort/snort.conf -i etho**

-A full →Alerts include full packet details: headers, payload, and metadata.

Adv:

Deep packet analysis.

Forensic investigations.

Alert Modes (-A)

- **Console Alert Mode**

Example: snort -A console -c /etc/snort/snort. conf -i etho

Sends alerts directly to the terminal screen.

- **Adv:**

Real-time view while Snort runs.

Real-time monitoring during testing or troubleshooting.

Alert Modes (-A)

UNIX Socket Mode

Example: snort -A unsock -c /etc/snort/snort.conf -i etho

-A unsock → Sends alerts to a UNIX socket instead of a file or terminal.

Adv:

Advanced setups with other security tools.

Used for external integrations (e.g., SnortSam, Guardian- *add-on tools* that extend **Snort's intrusion detection** capability into **intrusion prevention** by enabling *active response*.).

Snort Sam: A plugin/agent for Snort that allows it to communicate with firewalls and block malicious traffic dynamically.

Guardian: A Perl script that works with Snort's log/alert files to block attackers dynamically.

Alert Modes (-A)

No Alerts

- snort -A none -c /etc/snort/snort.conf -i etho
- → Disables alerting entirely.
- Snort still captures/logs packets if logging is enabled(-l)

Adv:

Packet capture only.

Performance testing without alert noise.

Binary Logging

- **-b(Binary Logging)**
  - Tells Snort to log packets in **binary (pcap) format**, rather than plain text.
  - **Example: snort -b -i etho -l /var/log/snort**
  - The **-b option in Snort** logs packets in **tcpdump (pcap) format**, which is:
  - **Fast** – Binary logging is quicker than writing text logs.
  - **Standardized** – The .pcap format is widely supported by tools like **tcpdump, Wireshark, Tshark, and Snort itself**.
  - **Lossless** – It stores the full raw packets (headers + payload), not just summaries.

Specify the configuration file

-c(configuration file)

The -c option in Snort is used to specify the configuration file (snort.conf) that Snort should use when running in IDS/IPS mode.

Example:snort -c /etc/snort/snort.conf -i etho

- **-c /etc/snort/snort.conf** → Uses the configuration file from /etc/snort/snort.conf.
- **-i etho** → Listens to network traffic on interface etho.
- When you run this, Snort:
- Reads all rules, variables (e.g., HOME_NET, EXTERNAL_NET), and preprocessors from snort.conf.
- Starts detecting intrusions based on those rules.

Daemon Mode

- **-D(daemon mode)**
- This option enables Snort to run in the background. In almost all implementations of Snort, this option is used. You don't use this option when you are testing Snort after installation.

Example: snort -c /etc/snort/snort.conf -i etho -D -l /var/log/snort

- **-c /etc/snort/snort.conf → Load the configuration file and rules.**
- **-i etho → Monitor network traffic on interface etho.**
- **-D → Start Snort in daemon mode (background).**
- **-l /var/log/snort → Log alerts and packet data to /var/log/snort.**

Network Interface

-I(Network Interface)

- This option is used to start Snort so that it listens to a particular network interface.
- This option is very useful when you have multiple network adapters and want to listen to only one of them.
- It is also useful when you want to run multiple Snort sessions on multiple network interfaces.
- For example, if you want Snort to listen to network interface eth1 only, you will use "-i eth1" on the command line while starting Snort.

Log Directory

-l (Log Directory)

- This option is used to set the directory where Snort logs messages.
- The default location is /var/log/snort.
- For example, if you want all log files to be generated under / snort directory, you will use "-l /snort" command line option.

Example: snort -i etho -c /etc/snort/snort.conf -l /snort

- **-i etho → Monitor the etho interface.**
- **-c /etc/snort/snort.conf → Load rules and configuration.**
- **-l /snort → Store logs inside /snort directory.**

SMB (Server Message Block) message pop-ups

-M(SMB (Server Message Block) message pop-ups)

- You have to specify a text file as argument to this option.
- The text file contains a list of Microsoft Windows hosts to which you want to send SMB pop-up windows.
- Each line should contain only one IP address.
- SMB runs on **TCP 445 port** – Modern SMB traffic directly over TCP/IP.

SMB (Server Message Block) message pop-ups

Suppose you have a file /etc/snort/smb_targets.txt with:

- 192.168.1.10
- 192.168.1.20
- 192.168.1.30

Run Snort with:

- snort -M /etc/snort/smb_targets.txt -i etho -c /etc/snort/snort.conf

Explanation:

- -M /etc/snort/smb_targets.txt → Specifies Windows hosts for pop-ups.
- -i etho → Monitor network traffic on etho.
- -c /etc/snort/snort.conf → Use Snort configuration and rules.

When Snort detects an event (like a malware signature or alert configured in rules), it can trigger a pop-up on the listed Windows hosts.

Testing the configuration

-T(testing the configuration)

- This option is very useful for testing and reporting on the Snort configuration.
- You can use this option to find any errors in the configuration files.

Example:snort -T -c /etc/snort/snort.conf

- -T → Test mode (no packet sniffing).
- -c /etc/snort/snort.conf → Specifies the configuration file to validate.

If everything is correct, Snort will output something like:

Snort successfully validated the configuration!

- If there are errors, Snort will show line numbers and error messages, so you can fix them before running Snort live.

SNORT MODES

SNORT MODES

- **2 Modes: Packet sniffer mode and NIDS mode**

Packet Sniffer Mode:

- It can be used as a packet sniffer, like tcpdump or snoop.
- When sniffing packets, Snort can also log these packets to a log file.
- The file can be viewed later on using Snort or tcpdump.
- No intrusion detection activity is done by Snort in this mode of operation.
- Using Snort for this purpose is not very useful as there are many other tools available for packet logging.
- **Command: snort -v → just packet capture & display.**

Packet Sniffer Mode

Command: snort -v

```
[root@conformix snort]# /opt/snort/bin/snort -v
Initializing Output Plugins!
Log directory = /var/log/snort

Initializing Network Interface eth0

        --== Initializing Snort ==--
Decoding Ethernet on interface eth0

        --== Initialization Complete ==--

-*> Snort! <*-
Version 1.9.0 (Build 209)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
11/20-15:56:14.632067 192.168.1.100:2474 -> 192.168.1.2:22
TCP TTL:128 TOS:0x0 ID:4206 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x9DAEEE9C  Ack: 0xF5683C3A  Win: 0x43E0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/20-15:56:14.632188 192.168.1.2:22 -> 192.168.1.100:2474
TCP TTL:64 TOS:0x10 ID:57042 IpLen:20 DgmLen:200 DF
***AP*** Seq: 0xF5683C8A  Ack: 0x9DAEEE9C  Win: 0x6330  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

- Timestamp
- Source/Destination IP and Ports
- Protocol (TCP/UDP/ICMP)
- TTL, TOS, Packet ID, Header Length, Payload Size
- TCP Flags, Sequence & Acknowledgement numbers, Window size

Packet Sniffer Mode

Command: snort –dv → to view application layer data

command still does not display all of the packet data.

```
[root@conformix snort]# /opt/snort/bin/snort -dv
Initializing Output Plugins!
Log directory = /var/log/snort

Initializing Network Interface eth0

        --== Initializing Snort ==--
Decoding Ethernet on interface eth0

        --== Initialization Complete ==--

-*> Snort! <*-
Version 1.9.0 (Build 209)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
11/20-16:18:11.129548 192.168.1.100:2474 -> 192.168.1.2:22
TCP TTL:128 TOS:0x0 ID:4387 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x9DAEF2FC  Ack: 0xF5688CDA  Win: 0x4190  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/20-16:18:11.129723 192.168.1.2:22 -> 192.168.1.100:2474
TCP TTL:64 TOS:0x10 ID:57171 IpLen:20 DgmLen:120 DF
***AP*** Seq: 0xF5688D2A  Ack: 0x9DAEF2FC  Win: 0x6330  TcpLen: 20
C5 1D 81 8F 70 B7 12 0B C1 1B 8F 6D A9 8F 1D 05  ....p......m....
40 7D F9 BD 84 21 11 59 05 01 E4 A1 01 20 AC 92  @}...!.Y..... ..
58 50 73 8D 17 EA E2 17 AD 3A AD 54 E2 50 80 CB  XPs......:.T.P..
DA E1 40 30 7B 63 0D 79 5A D8 51 07 93 95 2B A8  ..@0{c.yZ.Q...+.
F8 D4 F5 FA 76 D6 27 35 E8 6E E2 ED 41 2B 01 2D  ....v.'5.n..A+.-

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

- Timestamp
- Source ⟶ Destination ⟶
- TCP header info ⟶ TTL, TOS, ID, Seq, Ack, Win, Flags.
- Payload (data) ⟶ dumped in hex on the left, ASCII on the right.

Logging SNORT Data

- **1. Text Logging (ASCII Mode)**
- Stores packets in a **human-readable format**.
- By default, Snort writes logs to /var/log/snort/ (or the directory you specify).
- Each alert is logged in a plain text file (e.g., alert, packet.log).
- Good for **quick inspection**, but not efficient for **large-scale traffic**.

snort -c /etc/snort/snort.conf -l /var/log/snort/ -K ascii

-c ⟶ Snort config file.

-l ⟶ Log directory.

-K ascii ⟶ Log in plain text.

Output:

```
09/10-12:20:33.456789  [**] [1:1000001:0] ICMP test detected [**]
10.0.2.15 -> 10.0.2.255
ICMP TTL:64 TOS:0x0 ID:12345 IpLen:20 DgmLen:84
```

Logging SNORT Data

- **2. Binary Logging (pcap format)**
- Logs packets in **binary (tcpdump) format**.
- More efficient for storage and analysis.
- Cannot be read directly (need tcpdump or Wireshark to analyze).
- Useful for **forensics** and detailed investigation.

snort -c /etc/snort/snort.conf -l /var/log/snort/ -b

Output :

Creates files like /var/log/snort/snort.log.xxxxxxxxxx.

You can read them using:

tcpdump -r /var/log/snort/snort.log.1694338296

or open in **Wireshark**.

Logging SNORT Data

Feature	Text Logging (ASCII)	Binary Logging (pcap)
Format	Human-readable text	Compact binary (tcpdump format)
File size	Larger (verbose)	Smaller, efficient
Readability	Directly view with cat/less	Needs tcpdump/Wireshark to read
Use case	Quick inspection/debugging	Long-term storage, forensic analysis

SNORT MODES

NIDS Mode

- Works like tcpdump, captures and logs packets without intrusion detection.
- Logs can be viewed using Snort or tcpdump.
- In NIDS mode, Snort analyzes packets using rules instead of logging every packet.
- Only packets matching a rule trigger alerts or logs; others are silently dropped.

SNORT MODES

Starting Snort in NIDS Mode

- snort -c /opt/snort/etc/snort.conf
- Reads snort.conf and included files (rules, configs).
- After any configuration changes, restart Snort.

Logging and Running Options

- Log and display packets while in NIDS:
- snort -dev -l /var/log/snort -c /etc/snort/snort.conf
- For long-term monitoring, run in daemon mode (no console logging):
- snort -D -c /etc/snort/snort.conf
- Excessive logging increases disk and CPU usage.

SNORT ALERT MODES

1. Fast Mode

- Logs minimal alert info:
  - Timestamp
  - Alert message
  - Source & destination IP
  - Source & destination ports
- Command:

snort -c /opt/snort/etc/snort.conf -q -A fast

Logs to: /var/log/snort/alert.

SNORT ALERT MODES

2. Full Mode (Default)

- Logs detailed packet header info with the alert.
- Command:

snort -c /opt/snort/etc/snort.conf -q -A full

- Includes TTL, TOS, header length, packet length, ICMP type/code, etc.

SNORT ALERT MODES

3. UNIX Socket Mode

- Sends alerts to another program via UNIX sockets.
- Command:

snort -c /opt/snort/etc/snort.conf -a unsock

SNORT ALERT MODES

4. No Alert Mode

- Disables alerting completely.
- Useful for high-speed environments with unified logging.
- Command:
- snort -c /opt/snort/etc/snort.conf -A none

SNORT ALERT MODES

5. Syslog Mode

- Sends alerts to the Syslog daemon.
- Logs typically go to /var/log/messages.
- Command:

snort -c /opt/snort/etc/snort.conf -s

SNORT ALERT MODES

6. SNMP Trap Mode

- Sends alerts as SNMP traps to network management systems (e.g., OpenNMS, MRTG, HP OpenView).
- Supports SNMP v2 and v3.
- Configured via output plug-ins.

Sending Alerts to Windows

- Snort can send alerts as pop-up windows to Microsoft Windows machines using the Windows Messenger Service.
- Windows Messenger Service must be running on the target Windows machine.
  - Found under *Control Panel* → *Administrative Tools* → *Services*.

Running Snort in Stealth Mode

- Stealth mode hides the Snort sensor from the network, making it invisible to attackers.
- Running **Snort in stealth mode** means you want Snort to **listen to traffic without revealing itself** — typically by:

➢ **Not having an IP address on the interface** (so attackers can't ping or scan it).

➢ **Putting the interface into promiscuous mode** (so it can sniff all packets).

Running Snort in Stealth Mode

Steps to Run Snort in Stealth Mode

- **1. Disable IP on the Interface**
- You must configure the network interface card (NIC) used for sniffing **without an IP address**. Example (on Linux):

sudo ifconfig etho 0.0.0.0 up

- **ifconfig** → Legacy tool to configure network interfaces.
- **etho** → The network interface card (NIC) you're configuring.
- **0.0.0.0** → Assigns a null IP address (basically "no usable IP").
- **up** → Brings the interface into an active state (turned on).

→ This ensures the NIC is up and can capture traffic, but it doesn't have an IP, so it can't be reached over the network.

Running Snort in Stealth Mode

Step 1: Alternative: using ip command

- sudo ip addr flush dev etho

ip addr flush → Removes all IP addresses assigned to the given interface.

dev etho → Specifies the device (NIC).

→ This is the modern equivalent of removing any IP address. If your interface had an IP assigned earlier (via DHCP or static config), this command wipes it clean.

- sudo ip link set etho up

ip link set → Used to change the state of a network interface.

etho → The NIC to configure.

up → Activates the NIC.

→ After flushing the IP, this ensures the NIC is still up so that Snort (or tcpdump, Wireshark, etc.) can listen to traffic in promiscuous mode.

→ This way, the interface can capture traffic but won't respond to pings or connections.

Running Snort in Stealth Mode

Add Stealth Options

- Use **-D** to run Snort as a **daemon** (background, less visible).
- Avoid verbose outputs.

sudo snort -i etho -c /etc/snort/snort.conf -A fast -D

- **-i etho** → interface
- **-c /etc/snort/snort.conf** → configuration file
- **-A fast** → fast alerting mode (writes minimal alerts to log)
- **-D** → run in daemon (background) mode

Running Snort in Stealth Mode

Confirm Stealth Mode

- Run ifconfig or ip addr and check that **no IP is assigned** to the sniffing NIC.
- Use tcpdump -i etho in another terminal to ensure packets are being captured.
- Snort logs (usually in /var/log/snort/) should contain alerts.

→ With this setup, Snort will be **invisible on the network** (no IP, no ARP, no ping replies) but still capture and analyze all traffic.

WORKING WITH SNORT RULES

WORKING WITH SNORT RULES

- Intruders leave signatures (patterns in packet headers or payloads).
- Snort uses these signatures to create rules for detecting malicious activity.
- Signatures may be derived from:
  - Known vulnerabilities (databases of exploits).
  - Honeypots (trap systems that reveal attacker behavior).
- Rules are:
  - Easy to read (mostly one-line, can span multiple lines using \).
  - Stored in configuration files (snort.conf).
  - Can be organized across multiple files and included as needed.

TCP/IP Layers (Relevant to Snort)

- Snort primarily analyzes Layer 3 (Network) and Layer 4 (Transport), but may detect some anomalies in other layers.
- The 5 TCP/IP Layers:
- Physical Layer
  - Physical media, network interface adapter, drivers.
- Data Link Layer (Network Interface Layer)
  - Manages Ethernet (MAC) addresses.
- Network Layer (IP Layer)
  - Uses IP protocol (RFC 791).
  - Includes ICMP protocol (RFC 792).
  - Handles point-to-point communication.
- Transport Layer (TCP/UDP)
  - TCP (RFC 793): Reliable, connection-oriented.
  - UDP (RFC 768): Unreliable, connectionless.
- Application Layer
  - User-facing protocols: Telnet, HTTP, FTP, etc.

The First Bad Rule

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

Why is it bad?

- It alerts on every IP packet, filling up disk space quickly.
- It provides no useful information for intrusion detection.

- **alert** – Action to take (generate an alert).
- **ip** – Protocol (applies to all IP packets).
- **any (source IP)** – Match all source IPs.
- **any (source port)** – Match all source ports (irrelevant at IP layer).
- **->** – Direction of traffic.
- **any (destination IP)** – Match all destination IPs.
- **any (destination port)** – Match all destination ports (irrelevant at IP layer).
- **msg**: "..." – Message displayed when the rule triggers.

Another Simple Test Rule

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

- Alerts on all ICMP packets.
- Useful to test Snort using a simple ping.
- Example ping test:
- ping 192.168.2.1
- (Send from the Snort machine to a host or gateway in the same network.)

Structure of a Rule

- All Snort rules have two logical parts: rule *header* and rule *options*.

1. Rule Header

- Specifies what action the rule should take (alert, log, pass, etc.).
- Defines the criteria for matching packets (e.g., protocol, source, destination, ports).

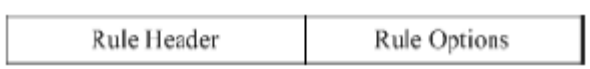


Figure 3-1 Basic structure of Snort rules.

2. Rule Options

- Usually includes:
  - Alert message (displayed when rule triggers).
  - Information on which part of the packet to inspect for generating the alert.
  - Additional matching criteria (payload, flags, content patterns, etc.).

Rule Headers, Rule Options

SNORT RULE HEADER AND OPTIONS

- The rule header contains the **criteria for matching packets** and **action to be taken**. Its components, in order, are:
- **Action**
  - Defines what happens when a packet matches the rule.
  - Common actions: alert, log, pass, or invoke another rule.
  - Example: alert generates an alert message and logs the packet by default.
- **Protocol**
  - Specifies which protocol the rule applies to (IP, ICMP, TCP, UDP, etc.).
  - Non-matching protocols are ignored to save CPU resources.
  - Example: icmp applies the rule only to ICMP packets.
- **Source Address & Port**
  - Defines the origin of the packet.
  - Can be a single host, multiple hosts, or network addresses.
  - Use any for all addresses/ports.
  - Ports are relevant only for TCP/UDP.

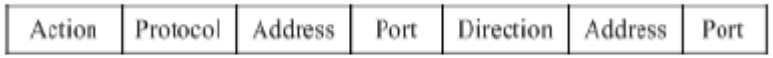


Figure 3-2 Structure of Snort rule header.

SNORT RULE HEADER AND OPTIONS

- **Direction**
  - Specifies packet flow: -> (left to right), <- (right to left), <> (either direction).
  - Determines which address/port is considered source or destination.
- **Destination Address & Port**
  - Defines the target of the packet.
  - Can use any for all addresses/ports.

Rule Options

- **The rule options, enclosed in parentheses (), define additional criteria and alert messages.**
- **Example:**
- **(msg: "Ping with TTL=100"; ttl: 100;)**
- **Explanation:**
  - msg: Text displayed when the rule triggers.
  - ttl: 100: Only matches packets where the IP header TTL field equals 100.

Component	Value	Meaning
Action	alert	Generate an alert and log the packet
Protocol	icmp	Apply only to ICMP packets
Source Address	any	All source addresses
Source Port	any	All ports (irrelevant for ICMP)
Direction	->	Packets flow from source to destination
Destination Address	any	All destinations
Destination Port	any	All ports (irrelevant for ICMP)
Options	(msg: "Ping with TTL=100"; ttl: 100;)	Alert message and match TTL=100 only

Snort Rule Actions

- **Predefined Actions**
- **Pass**
  - Ignores the packet.
  - Useful to skip known safe traffic or internal vulnerability assessment hosts, improving Snort performance.
- **Log**
  - Logs the packet to a file or database.
  - Level of detail depends on configuration and command-line arguments.
  - Does **not** generate an alert message.

Snort Rule Actions

- **Alert**
  - Generates an alert **and** logs the packet.
  - Alerts can be sent to console, file, or other output mechanisms.
  - Functional difference: Alert = message + log, while Log = only log.
- **Activate**
  - Generates an alert and **activates another rule** for further testing.
  - Useful when more complex, multi-step packet inspection is needed.
- **Dynamic**
  - Can only be triggered by another rule using the activate action.
  - Not applied to packets directly.

Snort Rule Actions

- **User-Defined Actions**
- **Send alerts to Syslog**
  - Syslog creates logs in /var/log.
  - Location configurable via /etc/syslog.conf.
- **Send SNMP traps**
  - Integrates alerts into Network Management Systems (NMS) like HP OpenView or OpenNMS.
- **Multiple actions on a packet**
  - Example: send an SNMP trap **and** log the alert to Syslog simultaneously.
- **Logging to XML files**
- **Database logging**
- Supports MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.

Defining a user-defined action in snort.conf:

```
ruletype action_name
{
   action definition
}
```

```
ruletype smb_db_alert
{
  type alert
  output alert_smb: workstation.list
   output database: log, mysql, user=rr password=rr dbname=snort host=localhost
 }
```

ruletype: Keyword to define a new action type.
action_name: Name of the user-defined action.
The block {} contains the action definition, similar to a function in C.

Protocols & Address

- The **protocol** part of a Snort rule specifies which type of packet the rule applies to.
- Snort currently supports these protocols:
  - **IP**
  - **ICMP**
  - **TCP**
  - **UDP**
- Protocol determines how Snort inspects the packet headers:
  - IP protocol: checks the link layer header.
  - Other protocols: uses the IP header to identify packet type.

- **Address**
- Snort rules have **two address fields**: source and destination.
- Addresses can be:
  - Single host (/32)
  - Class C network (/24)
  - Class B network (/16)
  - Class A network (/8)
  - Any subnet using CIDR notation (flexible number of bits)

Protocols & Address

Address	Meaning
192.168.1.3/32	Single host
192.168.1.0/24	Class C network
152.168.0.0/16	Class B network
10.0.0.0/8	Class A network
192.168.1.16/28	16-address subnet (14 usable)

Address Exclusion: Use ! to ignore specific addresses or subnets.

Example:

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

Example:
alert icmp ![192.168.2.0/24,192.168.8.0/24] any -> any any (msg: "Ping with TTL=100"; ttl: 100;)

Port Numbers

- Ports are relevant **only for TCP and UDP protocols**.
- Source and destination ports can be specified, or use any to match all ports.
- **Example: Alert for Telnet traffic containing "confidential" from a Class C network:**

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

- Direction modification: **<-> or <> can make rule apply in both directions.**

Port Ranges

- Specify with colon :
  - **1024:2048 → ports 1024 through 2048**
  - **:1024 → all ports up to 1024**
  - **1000: → all ports from 1000 and above**
- Negation: Use ! to exclude a port or range
- Example:

log udp any !53 -> any any (msg: "Logging all UDP except DNS";)

- Note: Commas cannot be used in port fields. Use ranges instead.
- Well-Known Ports: Standard ports for common applications (e.g., 80 for HTTP, 443 for HTTPS).

Direction

Symbol	Meaning	Example Use
->	Left side = source, Right side = destination	alert tcp any any -> 192.168.1.10 80 (msg: "HTTP access";) monitors traffic to the web server.
<-	Right side = source, Left side = destination	alert tcp any any <- 192.168.1.10 80 (msg: "HTTP access";) monitors traffic from the web server.
<>	Monitors traffic in both directions	alert tcp any any <> 192.168.1.10 23 (msg: "Telnet traffic";) monitors Telnet traffic to and from the server.

<> is especially useful for client-server applications, where you want to monitor **all traffic both ways**.

Snort rule keywords

Keyword	Purpose	Example
ack	Match TCP Acknowledgement Number (detect TCP ping scans)	alert tcp any any -> 192.168.1.0/24 any (flags:A; ack:0;)
classtype	Groups rules by attack type + priority Lower number = higher priority!	alert udp any any -> 192.168.1.0/24 6838 (msg:"DoS"; \ content: "server"; classtype:DoS;)
content	Search for string/pattern inside packet data Can use **ASCII** or **HEX** (|47 45 54| = GET). You can use the depth keyword to define the point after which Snort should stop	content:"GET"; alert tcp 192.168.1.0/24 any -> any any \ (content: "HTTP"; offset: 4; msg: "HTTP matched";) alert tcp 192.168.1.0/24 any -> any any (content: \ "HTTP"; offset: 4; depth: 40; msg: "HTTP matched";)
		content-list:"porn";
content-list	Load search patterns from external file	alert ip any any -> 192.168.1.0/24 any (content-list:"spam"; msg:"Spam word matched";)
dsize	Check packet data size (detect buffer overflow)	alert ip any any -> any any (dsize:>6000; msg:"Large packet";)

Feature	`content`	`content-list`
Matches	Single string	Multiple strings from a list
Defined in rule	Yes	Usually external list file
Flexibility	Less flexible for many keywords	Efficient for large keyword lists
Snort version	Works in Snort 2 & 3	Mainly Snort 3

Class Type	Description
`attempted-admin`	Attempt to gain administrative access
`attempted-user`	Attempt to gain normal user access
`bad-unknown`	Suspicious activity of unknown type
`denial-of-service`	DoS attack
`successful-admin`	Successful administrative compromise
`successful-admin`	Successful administrative compromise
`policy-violation`	Activity that violates network policy
`misc-activity`	Miscellaneous suspicious activity
`web-application-attack`	Attacks on web applications
`trojan-activity`	Trojan malware activity

Snort rule keywords

Keyword	Purpose	Example
flags	Match TCP flags (SYN, FIN, ACK, etc.) Can combine with ! (NOT), + (AND), * (OR).	alert tcp any any -> 192.168.1.0/24 any (flags:SF; msg:"SYN-FIN scan detected";)
fragbits	D = Don't Fragment, M = More Fragments, R = Reserved. Match IP fragmentation bits (D, M, R)	fragbits:D; alert icmp any any -> any any (fragbits:D; msg:"DF set";)
icmp_id / icmp_seq	Match ICMP identifier or sequence	alert icmp any any -> any any (icmp_id: 100; \ msg: "ICMP ID=100";)
itype / icode	Matches ICMP **type** (e.g., Echo Request = 8, Echo Reply = 0). Matches ICMP **code** (explains type in detail, e.g., redirect reasons).	alert icmp any any -> any any (itype:5; icode:1; msg:"Host redirect detected";)
id	Match IP fragment ID	id:12345;

Table 3-2 TCP flag bits

Flag	Argument character used in Snort rules
FIN or Finish Flag	F
SYN or Sync Flag	S
RST or Reset Flag	R
PSH or Push Flag	P
ACK or Acknowledge Flag	A
URG or Urgent Flag	U
Reserved Bit 1	1
Reserved Bit 2	2
No Flag set	0

Table 3-3 ICMP type filed values

Value	Type of ICMP Packet
0	Echo reply
3	Destination unreachable
4	Source quench
5	Redirect
8	Echo request
11	Time exceed
12	Parameter problem
13	Timestamp request
14	Timestamp reply
15	Information request
16	Information reply

Snort rule keywords

Keyword	Purpose	Example
	Match IP header options (rr, ts)	
ipopts	rr → Record Route ts → Timestamps lsrr → Loose Source Routing ssrr → Strict Source Routing	alert ip any any -> any any (ipopts: lsrr; msg:"Loose source routing attempt";)
ip_proto	Match IP protocol number	alert ip any any -> any any (ip_proto: 94; msg:"IPIP tunneling";)
logto	Log packets to specific file	alert icmp any any -> any any (logto:mylog; ttl:100;)
msg	Add custom message to alert/log	msg:"TCP SYN Scan";
nocase	Make content search case-insensitive	content:"get"; nocase;

Protocol	IP Proto Number
ICMP	1
TCP	6
UDP	17
GRE	47
ESP	50

Snort rule keywords

Keyword	Purpose	Example
priority	Assign severity level (1 = high)	priority:1;
react	Block or warn sessions (needs special build)	alert tcp 192.168.1.0/24 any -> any 80 (msg:"Block HTTP"; react:block;)
reference	Add external reference (CVE, Bugtraq)	reference:cve,2001-0876;
resp	Send reset/ICMP to disrupt attacker rst_snd → Reset to sender rst_rcv → Reset to receiver rst_all → Reset both sides icmp_net → ICMP network unreachable	alert tcp any any -> 192.168.1.0/24 8080 (resp:rst_snd;)
rev	Rule revision number	rev:2;

Snort rule keywords

Keyword	Purpose	Example
rpc	Detect RPC requests (app, proc, version)	alert ip any any -> any any (rpc:10000,*,3; msg:"RPC request";)
sameip	Detect spoofing (src = dst IP)	alert ip any any -> any any (msg:"Possible spoofed packet - source and destination IP are the same"; sameip; sid:1000001; rev:1;)
seq	Match TCP sequence number	alert tcp any any -> any any (msg:"TCP packet with sequence number 1000"; seq:1000; sid:2000001; rev:1;)
flow	Match TCP session direction/state	alert tcp any any -> any 80 (msg:"Established HTTP session traffic to server"; flow:to_server,established; sid:2000002; rev:1;)
session	Dump TCP session data	log tcp any any -> 192.168.1.0/24 110 (session:printable;)

Snort rule keywords

Keyword	Purpose	Example
sid	Unique Snort rule ID	0–99 reserved 100–1,000,000 Snort rules >1,000,000 local rules Example: sid:1000001;
tag	Log extra packets after trigger	tag:session,100,packets;
tos	Match Type of Service (TOS) value	tos:1;
ttl	Match TTL in IP header (detect traceroute)	ttl:100;
uricontent	Search only HTTP URI part	uricontent:"/admin";

THE SNORT CONFIGURATION FILE

THE SNORT CONFIGURATION FILE

- Snort needs a configuration file at startup.
- Default name = snort.conf (but you can use any name).
- Run with:
- snort -c /path/to/snort.conf
- Alternative: .snortrc in your home directory.
- Advantage: You can run multiple Snort instances on different interfaces with different configs.

Sections of snort.conf

- **Variable Definitions** → store values like networks, ports, file locations.
- **Config Parameters** → general Snort settings (can also be given via command line).
- **Preprocessor Config** → preprocess traffic before the detection engine.
- **Output Modules** → control how Snort logs alerts/data.
- **New Action Types** → define custom actions (if default ones are not enough).
- **Rules & Include Files** → rules usually kept in separate files and included into snort.conf using include.

Using Variables in Rules

- Variables make rules easier to manage.
- Example:
- var HOME_NET 192.168.1.0/24
- alert ip any any -> $HOME_NET any (ipopts: lsrr; msg:"Loose source routing attempt"; sid:1000001;)
- Benefit: Only change variable values when moving Snort to another network, not every rule.

Variables as Lists

- Variables can contain multiple networks.
- Example:
- var HOME_NET [192.168.1.0/24,192.168.10.0/24]

Variables with Interface Names

- Variables can be based on interface IPs.
- Example (Linux):
- var HOME_NET $etho_ADDRESS
- var EXTERNAL_NET $eth1_ADDRESS
- Benefit: If interface IPs change → rules adapt automatically.

Using any

- any can be a variable.
- Matches everything (all IPs or ports).
- Example:
- var EXTERNAL_NET any

Table 3-6 Snort config directives

Directive	Description
order	Changes the order in which rules are applied. It is equivalent to the –o command line option.
alertfile	Used to set the name of the alert file. Alert file is created in log directory (see logdir directive).
classification	Builds classification for rules. See explanation of the classtype keyword used in rules.
decode_arp	Equivalent to –a command line option. It turns ON arp decoding.
dump_chars_only	Equivalent –C command line option.
dump_payload	Equivalent to –d command line option. It is used to dump the data part of the packet.
decode_data_link	Equivalent to –e command line option. Using this directive you can decode data link layer headers (Ethernet header, for example).
bpf_file	Equivalent to –F command line option.
set_gid	Equivalent to –g command line option. Using this directive you can set the group ID under which Snort runs. For example, you can use "config set_gid: mygroup"
daemon	Equivalent to –D command line option. It invokes Snort as daemon instead of foreground process.
reference_net	Equivalent to –h command line option. It sets the home network address.
interface	Equivalent to –i command line option. It sets the interface for Snort.
alert_with_interface_name	Equivalent to –T command line option. This directive is used to append the interface name to the alert message. This is sometimes useful if you are monitoring multiple interfaces on the same sensor.
logdir	Equivalent to –l command line option. It sets the directory where Snort logs data. The default location of the log directory is `/var/log/snort`.

Automatically Updating Snort Rules

✅ Simple Method (Shell Script)	✅ Sophisticated Method (Oinkmaster)
Uses wget to download rules archive. Steps: Download from Snort website. Extract rules (tar -zxf). Backup old rules. Move new rules into /etc/snort. Restart Snort (/etc/init.d/snortd restart).	Tool written in Perl. Features: Downloads latest rules. Updates only changed rules. Can skip customized files (to protect your edits). Can disable specific rules permanently. Config file = oinkmaster.conf. Useful for automated updates (via cron).

Default Snort Rules & Classes

- Snort ships with many predefined rule files, each for a category.
- Examples:
  - dns.rules → DNS attacks
  - telnet.rules → Telnet port attacks
  - x11.rules → X-Windows attacks
  - backdoor.rules, ddos.rules, web-attacks.rules, etc.
- local.rules → for admin's own custom rules.

Checking su Attempts (Telnet)

- alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any \ (msg:"TELNET Attempted SU from wrong group"; \ flow:from_server,established; content:"to su root"; nocase; \ classtype:attempted-admin; sid:715; rev:6;)
- Triggers when someone tries “su root” inside a Telnet session.
- Key points:
  - $TELNET_SERVERS & $EXTERNAL_NET are variables.
  - Works on established server → client traffic.
  - Case insensitive (nocase).
  - Rule ID = 715, revision = 6.

Checking Incorrect Logins (Telnet)

- alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any \ (msg:"TELNET login incorrect"; content:"Login incorrect"; \ flow:from_server,established; reference:arachnids,127; \ classtype:bad-unknown; sid:718; rev:6;)
- Detects failed login attempts on Telnet.
- Includes reference to vulnerability database (arachnids).

This rule **alerts when a Telnet server sends "Login incorrect" to an external host,** indicating a failed login attempt. It only triggers for established connections from the server. The alert is classified as **unknown bad activity,** and references a threat entry in the **arachnids database**.

Writing Good Rules

- Best practices for strong Snort rules:
- Always include a msg (description).
- Use classtype to categorize rule.
- Assign a unique SID.
- Add reference (CVE, bugtraq, etc.).
- Use rev to track versions.
- Write generalized rules that detect variations of attacks.

PREPROCESSORS AND OUTPUT MODULES

PREPROCESSORS AND OUTPUT MODULES

- Purpose: Process packets before Snort rules are applied.
- Configured in snort.conf with keyword preprocessor.
- All enabled preprocessors check every packet (too many = slower Snort).
- You can even write your own preprocessors.

Original HTTP Request	What Attacker Does	Normalized Request
`GET /admin.php`	Uses URL encoding: `/admin%2Ephp`	`/admin.php`
`GET /login`	Inserts extra slashes: `/lo//gin`	`/login`
`GET /user`	Unicode encoding: `/u%0073er`	`/user`

1. HTTP Decode
Normalizes HTTP requests.
Stops attackers from hiding malicious URIs with hex encoding.
Example:
preprocessor http_decode: 80 8080 443

2. Portscan
Detects scanning attempts (first step of intrusions).
Example:
preprocessor portscan: 192.168.1.0/24 5 10
/var/log/snort/portscan.log
→ Logs if 5 ports scanned in 10 sec.
Types of scans: TCP connect, SYN, NULL, FIN, XMAS, UDP.
Ignore trusted hosts:
preprocessor portscan-ignorehosts: 192.168.1.10/32

5. SPADE (Statistical Packet Anomaly Detection Engine)
Detects unusual/anomalous packets.
Needs memory & processing power.

3. frag2
Defragments IP packets.
Default: 4MB memory, 60 sec timeout.
Example:
preprocessor frag2: 2097152, 30
The **frag2** preprocessor **reassembles fragmented IP packets**.
Snort can handle up to **2 MB of fragments** at a time.
If all fragments of a packet are not received within **30 seconds**, they are discarded.

4. stream4
Functions:
- TCP stream reassembly
- Stateful inspection

Modules: stream4 & stream4_reassemble.
Example options: timeout, memcap, ports (21, 23, 25, 53, 80, etc.).
preprocessor telnet: ports { 23 } memcap 131072 timeout 60
Monitors Telnet on port 23.
Uses 128 KB memory for session tracking.
Times out idle Telnet sessions after 60 seconds.

6. ARP Spoofing
Detects ARP anomalies (used for redirection attacks).
Example:
preprocessor arpspoof: -unicast
preprocessor arpspoof_detect_host: 192.168.1.13
34:45:fd:3e:a2:01

Common output types

- Database (MySQL, Oracle, PostgreSQL).
- SNMP traps (to monitoring systems).
- SMB alerts (Windows pop-up).
- Syslog (central logging).
- XML / CSV files (for parsing/import).

To log into MySQL:

- output database: log, mysql, user=rr password=rr dbname=snort host=localhost
- To send alerts as SMB pop-ups:
- output alert_smb: workstation.list

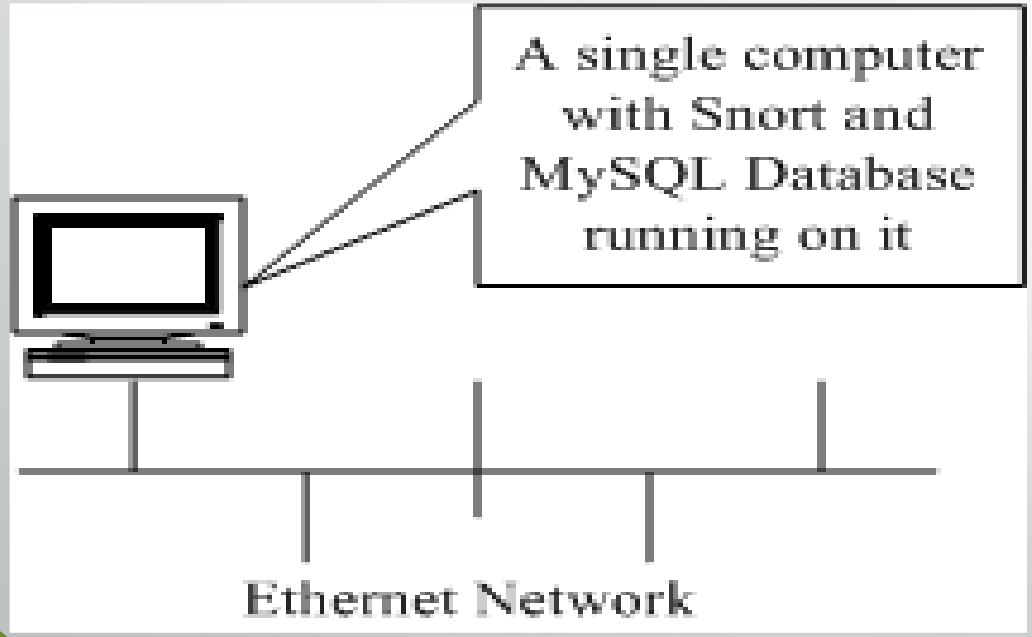
Common Output Modules

- alert_syslog → send alerts to syslog.
- alert_full → detailed logging (slower).
- alert_fast → quick one-line alerts (for high-speed networks).
- alert_smb → pop-up on Windows via Samba.
- log_tcpdump → store logs in tcpdump format.
- XML → logs in XML format (web-friendly).
- CSV → logs in spreadsheet-friendly format.
- Unified Logging → fast binary logging (used with Barnyard).
- SNMP Traps → send alerts to SNMP managers.
- Log Null → ignores alerts (not recommended).

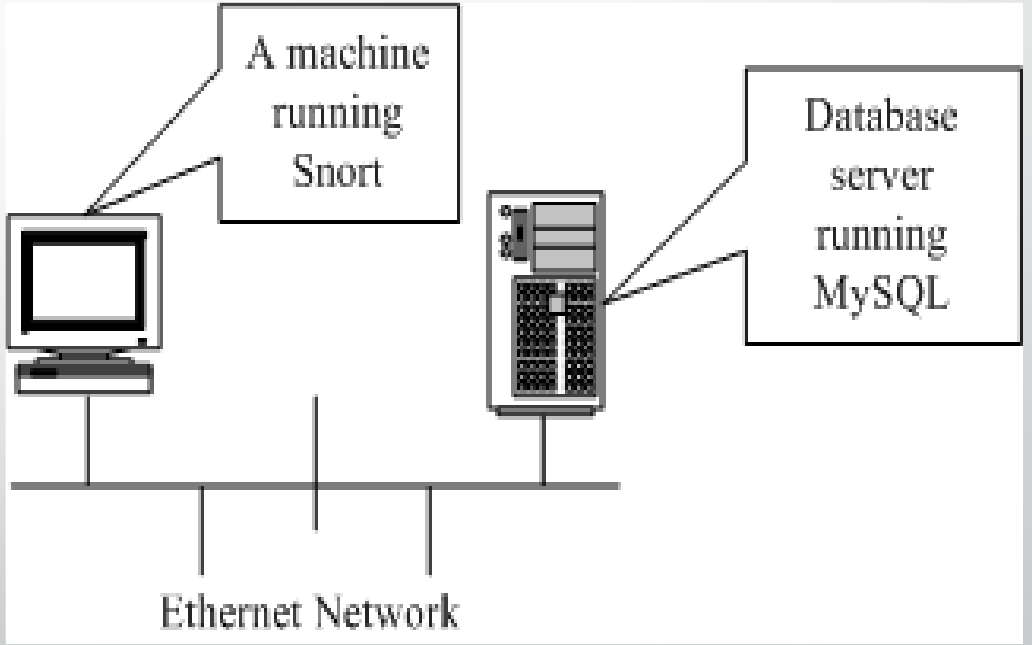
USING SNORT WITH MYSQL

USING SNORT WITH MYSQL

- You can install and run the MySQL database server on the same machine where Snort is running, as shown in Figure below.

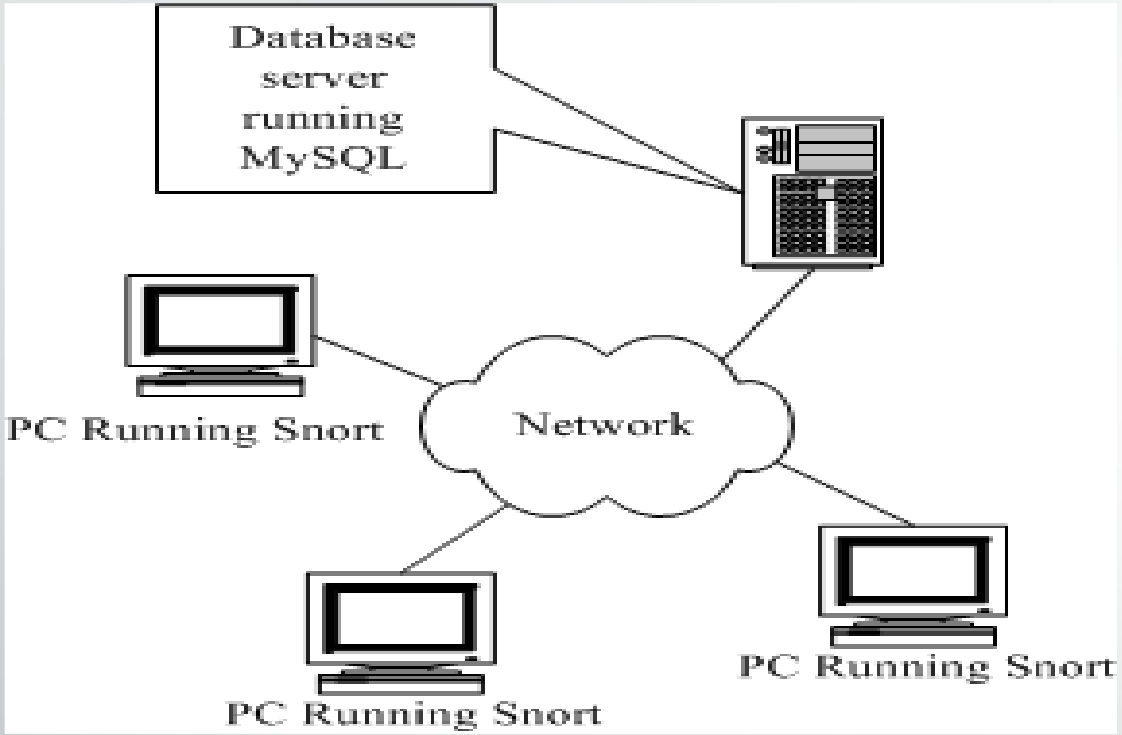


A single computer running Snort and MySQL database server.



A computer running Snort logging to a separate MySQL database server.

USING SNORT WITH MYSQL



Many Snort PCs logging data to a centralized MySQL database server.

- In a simple configuration, **both Snort and MySQL run on the same machine.**
- In larger networks, **Snort logs to a remote MySQL server.**
- In enterprise setups, **multiple Snort sensors send their logs to a centralized database server**, which acts as the central logging point.

Setting Up Snort with MySQL

- **Step 1** – Compile Snort with MySQL Support
  To enable database logging, Snort must be compiled with MySQL libraries using the --with-mysql option. For example:
- ./configure --prefix=/opt/snort --with-mysql=/usr/lib/mysql

- **Step 2** – Install MySQL
  MySQL should be installed either via system packages or by downloading from the official MySQL site.
- **Step 3** – Create Database
  Inside MySQL, create a dedicated database for Snort, usually named snort:
- **create database snort;**
- **use snort;**

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

mysql> use snort
Database changed
mysql> status
--------------
mysql  Ver 11.13 Distrib 3.23.36, for redhat-linux-gnu (i386)

Connection id:          41
Current database:       snort
Current user:           root@localhost
Current pager:          stdout
Using outfile:          ''
Server version:         3.23.36
Protocol version:       10
Connection:             Localhost via UNIX socket
Client characterset:    latin1
Server characterset:    latin1
UNIX socket:            /var/lib/mysql/mysql.sock
Uptime:                 1 hour 56 min 29 sec

Threads: 1  Questions: 107  Slow queries: 0  Opens: 14  Flush
tables: 1  Open tables: 7 Queries per second avg: 0.015
--------------
mysql>
```

The command "mysql -h localhost -u root –p " is used to **connect mysql client to a database server running on localhost.** The **"-u root" part shows the database user name** used to connect to the database. The **"-p"part is used to enter user password on the next line**. A welcome message is displayed after login and you get the "mysql>" prompt where you can issue other commands.

To end the mysql client session, you can use the "exit" command at the MySQL prompt.

- **Step 4** – Create User and Permissions
  A user account must be created for Snort with sufficient privileges (CREATE, INSERT, DELETE, UPDATE, SELECT). For example:
- grant CREATE,INSERT,DELETE,UPDATE,SELECT on snort.* to rr@localhost;
- set password for rr = password('rr78x');

- **Step 5** – Create Tables
  Snort provides an SQL script in the contrib directory. Running this script builds the necessary tables inside the snort database:
- mysql -u rr -p snort < contrib/create_mysql
- This creates around 16 tables, such as event, iphdr, tcphdr, and udphdr, each storing different parts of packet and alert information. Extra tables like protocols, services, and flags can also be added by running the snortdb-extra script.
- The following command uses this script to create all database tables in the snort database.
- [root@laptop]# mysql -h localhost -u rr -p snort < contrib/ create_mysql
- Enter password:
- [root@laptop]#

- Different command line options are used with this command.
- • The “-h localhost” part of the command is used to tell the mysql client that the database server is running on the same machine as the client.
- • The “-u rr” part is used to specify database user name to log into the database server. This is the same user that you created previously.
- • The “-p” part shows that you will enter the password for user rr in the next line.
- The “snort” part of the command line shows that the database that will be used to create tables is “snort.”
- The last part “<contrib./create_mysql” specifies a file name and shows that mysql client will read commands from this file.

```
[root@laptop]# mysql -h localhost -u rr -p snort
Enter password:
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 46 to server version: 3.23.36

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


mysql> show tables;

+------------------+
| Tables_in_snort  |
+------------------+
| data             |
| detail           |
| encoding         |
| event            |
| icmphdr          |
| iphdr            |
| opt              |
| reference        |
| reference_system |
| schema           |
| sensor           |
| sig_class        |
| sig_reference    |
| signature        |
| tcphdr           |
| udphdr           |
+------------------+
16 rows in set (0.00 sec)
```

- Step 6 – Configure Snort
  The Snort configuration file snort.conf must be modified to log to the database using an output plugin. For example:
- output database: log, mysql, user=rr password=rr78x dbname=snort host=localhost
- Step 7 – Start Snort
  Snort is then started with this configuration:
- /opt/snort/bin/snort -c /etc/snort/snort.conf
- Step 8 – Test the Database
  Once Snort is running, generate test traffic (such as ICMP pings or port scans) and verify that events are inserted into the database by checking tables like icmphdr or signature.

- 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";)
- To test these rules, we use the following two commands on a Microsoft Windows machine. I have used Windows XP Home Edition for the sake of experiment.
- ping -n 1 -f 192.168.1.2
- ping -n 1 -j 192.168.1.2 192.168.1.2

- The first command sends an ICMP echo packet with the don’t fragment (DF) bit set and thus triggers the second rule.
- The second command sends an ICMP packet with Loose Source Record Routing (lsrr) option set, which triggers the first rule.
- Both of these commands create alert messages.
- The alert messages are recorded in the database as you can see in different tables.
- For example, the icmphdr table contains ICMP headers corresponding to these alert messages.

```
mysql> select * from icmphdr;
+-----+-----+-----------+-----------+-----------+---------+----------+
| sid | cid | icmp_type | icmp_code | icmp_csum | icmp_id | icmp_seq |
+-----+-----+-----------+-----------+-----------+---------+----------+
|   1 |   1 |         8 |         0 |     18780 |    NULL |     NULL |
|   1 |   2 |         0 |         0 |     20828 |    NULL |     NULL |
|   1 |   3 |         8 |         0 |     18524 |    NULL |     NULL |
+-----+-----+-----------+-----------+-----------+---------+----------+
3 rows in set (0.00 sec)
```

Using Stunnel for Secure Remote Logging

- When Snort logs are sent to a remote MySQL server, they normally travel in plain text, which is insecure. Stunnel provides a solution by creating an encrypted SSL tunnel for communication.
- **On the server side, Stunnel listens on a different port (e.g., 3307) and forwards traffic securely to MySQL's standard port (3306):**
- stunnel -P/tmp/ -p stunnel.pem -d 3307 -r localhost:3306
- **On the Snort client machine, Stunnel is run in client mode to establish the secure connection:**
- stunnel -P/tmp/ -c -d 3306 -r SERVER_NAME:3307
- This setup ensures that all Snort logs are transmitted securely across the network, preventing interception or manipulation.

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

1️⃣ **stunnel**

This runs the **stunnel program,** which creates a **secure SSL/TLS tunnel** for connections.

2️⃣ **-P /tmp/**

-P specifies the **PID (process ID) file directory**.

Stunnel will write its PID file to /tmp/ so the process can be tracked or stopped later.

-p stunnel.pem

-p specifies the **certificate and private key file**.

stunnel.pem contains both:

- The **server certificate** (to prove identity).
- The **private key** (used for SSL encryption).

✅ This is essential because stunnel provides SSL/TLS encryption.

4️⃣ **-d 3307**

-d specifies the **local port** on which stunnel listens for incoming SSL/TLS connections.

Here, stunnel will **listen on port 3307** on the local machine.

5️⃣ **-r localhost:3306**

-r specifies the **remote server and port** to which stunnel will forward traffic after decrypting SSL.

Here:

- localhost → the MySQL server is running locally.
- 3306 → the standard MySQL port.

stunnel listens on local port 3307 for SSL-encrypted connections.

Incoming connections to 3307 are decrypted using stunnel.pem.

Traffic is forwarded unencrypted to the local MySQL server on port 3306.

This allows clients to connect securely via SSL without changing MySQL itself.

Database Optimization

- Snort generates large volumes of alerts, causing the MySQL database to grow quickly. Regular maintenance is essential.
- Use **OPTIMIZE TABLE** data; to defragment and improve performance (can be automated with cron jobs).
- For long-term management, **archive old data by either recreating the database or moving old records to archive tables.**
- Proper maintenance keeps Snort efficient and able to handle heavy data loads.