R.M.K GROUP OF ENGINEERING INSTITUTIONS

R.M.K
GROUP OF INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

Please read this disclaimer before proceeding:

This document is confidential and intended solely for the educational purpose of RMK Group of Educational Institutions. If you have received this document through email in error, please notify the system manager. This document contains proprietary information and is intended only to the respective group / learning community as intended. If you are not the addressee you should not disseminate, distribute or copy through e-mail. Please notify the sender immediately by e-mail if you have received this document by mistake and delete this document from your system. If you are not the intended recipient you are notified that disclosing, copying, distributing or taking any action in reliance on the contents of this information is strictly prohibited.

22CY701
INTRUSION DETECTION AND INTERNET SECURITY
(Lab Integrated)

UNIT II
INTRUSION DETECTION AND PREVENTION TECHNIQUES

Department : **CSE(CS)**

Batch/Year : **2022 - 2026 /IV**

Created by :**Dr. Dharini N**

Date : **10.08.2025**

R.M.K GROUP OF INSTITUTIONS

Table of Contents

S NO	CONTENTS	PAGE NO
1	Contents	1
2	Course Objectives	6
3	Pre Requisites (Course Names with Code)	8
4	Syllabus (With Subject Code, Name, LTPC details)	10
5	Course Outcomes	13
6	CO- PO/PSO Mapping	15
7	Lecture Plan	16
8	Activity Based Learning	19
9	Lecture Notes	21
	Lab Exercises	59
	Lecture Slides	90
10	Assignments	90
11	Part A (Q & A)	93
12	Part B Qs	98
13	Supportive Online Certification Courses	100
14	Real time Applications in day to day life and to Industry	102
15	Contents Beyond the Syllabus	104
16	Assessment Schedule	106
17	Prescribed Text Books & Reference Books	107
18	Mini Project Suggestions	109

Course Objectives

22CY701 INTRUSION DETECTION AND INTERNET SECURITY

(Lab Integrated)

COURSE OBJECTIVES

- To Understand when, where, how, and why to apply Intrusion Detection tools and techniques in order to improve the security posture of an enterprise.
- To Apply knowledge of the fundamentals and history of Intrusion Detection in order to avoid common pitfalls in the creation and evaluation of new Intrusion Detection Systems
- To Analyze intrusion detection alerts and logs to distinguish attack types from false alarms
- To Understand the fundamentals of network security, including the MAC layer, Internet Protocol, and common attacks targeting these layers.
- To Gain an understanding of key internet security mechanisms, including firewalls, virtual private networks (VPNs), and TLS/SSL VPNs.

Prerequisite

22CY701 INTRUSION DETECTION AND INTERNET SECURITY

PREREQUISITE

1. 22CY401-CYBER SECURITY ESSENTIALS
2. 22CS501 – COMPUTER NETWORKS
3. 22CS901 – ETHICAL HACKING

Syllabus

R.M.K
GROUP OF
INSTITUTIONS

22CY701 – INTRUSION DETECTION AND INTERNET SECURITY (Lab Integrated)

SYLLABUS **3 0 2 4**

UNIT I INTRODUCTION TO INTRUSION DETECTION

History of Intrusion detection, Audit, Concept and definition, Internal and external threats to data, attacks, Need and types of IDS, Information sources Host based information sources, Network based information sources.

List of Exercise/Experiments

1. Install Snort and configure it to monitor network traffic.

2. Deploy Snort as a Network Intrusion Detection System (NIDS).

UNIT II INTRUSION DETECTION AND PREVENTION TECHNIQUES

Intrusion Prevention Systems, Network IDs protocol based IDs , Hybrid IDs, Analysis schemes, thinking about intrusion. A model for intrusion analysis, techniques Responses requirement of responses, types of responses mapping responses to policy Vulnerability analysis, credential analysis non credential analysis

List of Exercise/Experiments

1. Write and implement custom Snort rules to detect specific traffic patterns.

2. Integrate Snort with MySQL to log alerts to a database.

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

List of Exercise/Experiments

1.Enhance Snort's functionality using preprocessors and plugins.

2.Set up advanced alerting and logging mechanisms.

UNIT IV ESSENTIALS OF INTERNET SECURITY

Network Security basics-The MAC Layer and Attacks- The Internet Protocol and Attacks- Packet Sniffing and Spoofing-Attacks on TCP Protocol- DNS Attacks Overview - Local DNS Cache Poisoning Attack- Remote DNS Cache Poisoning attack- Replay forgery attacks-DNS Rebinding attack- DoS on DNS Servers- DNSSEC-Securing DNS

List of Exercise/Experiments

1.Conducting TCP SYN Flood Attack

2.Sniffing Packets on Network Interfaces

3.Spoofing Source IP Address in Packets

4.Investigating DNSSEC Implementation and Validation

UNIT V INTERNET SECURITY MECHANISMS

Firewall-Virtual Private Network-Overview of How TLC/SSL VPN Works- Creating and using the TUN Interface- Implementing the IP Tunnel- Testing VPN- Tunneling and Firewall Evasion- -BGP and Attacks- The Heartbleed bug and attack- Reverse Shell

List of Exercise/Experiments

1. Configuring Linux Firewall using IP tables
2. Setting Up VPN Tunnels
3. Exploring BGP Session Hijacking
4. Simulating Heartbleed Attack Scenario

Course Outcomes

R.M.K GROUP OF INSTITUTIONS

COURSE OUTCOMES

- CO1: Understand fundamental concepts and demonstrate skills in capturing and analyzing network packets.
- CO2: Utilize various protocol analyzers and Network Intrusion Detection Systems (NIDS) to detect network attacks and troubleshoot network problems.
- CO3: Develop the ability to proficiently use the Snort tool for detecting and mitigating network attacks
- CO4: Demonstrate knowledge of network security basics, including MAC layer vulnerabilities and attacks, as well as common attacks targeting the Internet Protocol.
- CO5: Demonstrate understanding of firewall, VPN, and TLS/SSL VPN principles and functionalities in network security.
- CO6: Apply the concepts of Intrusion Detection and internet security protocols to develop cyber security mechanisms.

CO – PO/ PSO Mapping

R.M.K GROUP OF INSTITUTIONS

CO-PO MAPPING

COs	PO's/PSO's														
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	3	1	2	2	–	–	–	–	1	–	2	2	3	1
CO2	3	3	1	2	2	–	–	–	–	1	–	2	2	3	2
CO3	2	2	2	1	3	–	–	1	1	2	1	2	1	3	3
CO4	3	2	1	1	1	–	–	2	–	1	–	3	1	3	1
CO5	3	2	1	1	2	1	–	2	1	2	1	2	1	3	2
CO6	2	2	3	2	3	1	–	2	2	2	2	3	3	3	3

1 – Low, 2 – Medium, 3 – Strong

Lecture Plan

R.M.K
GROUP OF
INSTITUTIONS

LECTURE PLAN

S No	Topics	No of periods	Proposed date	Actual Lecture Date	Pert aining CO	Taxo nomy level	Mode of delivery
1	Introduction to Snort	1			CO3	K1	ICT Tools
2	Snort Installation Scenarios	1			CO3	K2	ICT Tools
3	Installing Snort	1			CO3	K3	ICT Tools
4	Running Snort on Multiple Network Interfaces	1			CO3	K2	ICT Tools
5	Snort Command Line Options. Step-By-Step Procedure to Compile and Install Snort Location of Snort Files	1			CO3	K2	ICT Tools
6	Snort Modes Snort Alert Modes	1			CO3	K2	ICT Tools
7	Working with Snort Rules, Rule Headers	1			CO3	K2	ICT Tools
8	Rule Options, The Snort Configuration File etc.	1			CO3	K2	ICT Tools
9	Rule Options, The Snort Configuration File etc.	1			CO3	K3	ICT Tools
10	Plugins, Preprocessors and Output Modules	1			CO3	K3	Lecture and Practical
11	Plugins, Preprocessors and Output Modules	1			CO3	K3	Lecture and Practical
12	Enhance Snort's functionality using preprocessors and plugins.	1			CO3	K3	Lecture and Practical
13	Enhance Snort's functionality using preprocessors and plugins.	1			CO3	K3	Lecture and Practical

14	Set up advanced alerting and logging mechanisms.	1			CO3	K3	Lecture and Practical
15	Set up advanced alerting and logging mechanisms.	1			CO3	K3	Lecture and Practical

Activity Based Learning

SNORT INSTALLATION AND CONFIGURATION

Lecture Notes

R.M.K GROUP OF INSTITUTIONS

UNIT III

1. INTRODUCTION

Snort can be installed as:

1. **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).
        - **Windows machine** via SMB pop-up.
2. **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.

A comprehensive working Snort system utilizes these tools to provide a web-based user interface with a backend database.

- 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-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 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 is used by ACID to connect to MySQL database.

2. 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.
  - 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.

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.

Single Sensor with Network Management Integration

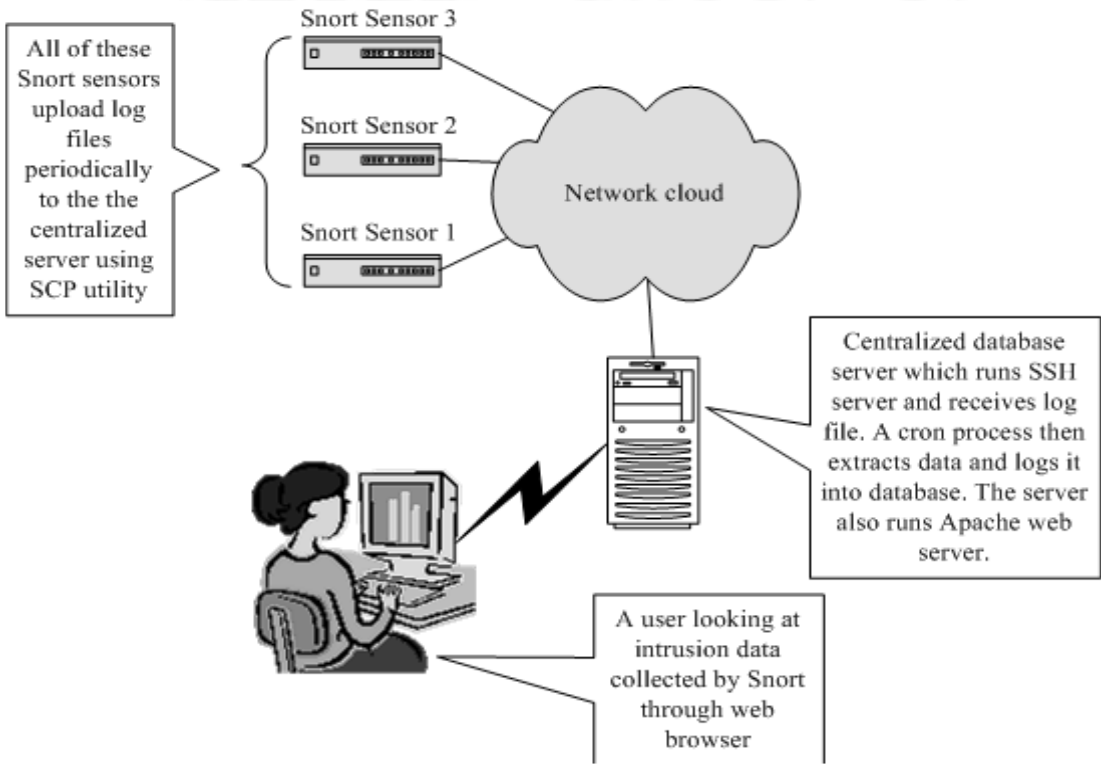
- **Purpose:** Integrates Snort with enterprise network management systems.
- **Supported Systems:** Hewlett-Packard, IBM, Computer Associates.
- **Integration:**
  - Uses **SNMP traps** to send alerts.
  - Requires SNMP capability during Snort compilation.

Single Sensor with Database & Web Interface

- **Purpose:** Easier analysis and management of Snort data.
- **Key Components:**
  - Snort sensor
  - Database server (MySQL, PostgreSQL, Oracle, MS SQL, etc.)
  - Web server (PHP used for database connection and web pages)
- **Features:**
  - Logs intrusion data to a database.
  - Web interface allows easy viewing and analysis of alerts.
  - Can run all components on the same system.
  - Requires username, password, and database details for setup.

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.



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

3. INSTALLING SNORT

Snort can be installed using precompiled packages (RPM) or by compiling from source code.

Installation from RPM Package

- **Download:**
  - Get the latest RPM from www.snort.org (e.g., snort-1.9.0-1snort.i386.rpm).
- **Install Command:**
- rpm --install snort-1.9.0-1snort.i386.rpm
  - Creates /etc/snort (rules/config files).
  - Creates /var/log/snort (log files).
  - Creates /usr/share/doc/snort-* (documentation).
  - Adds Snort daemon /usr/sbin/snort-plain.
  - Adds startup/shutdown script /etc/init.d/snortd.
- **Starting & Stopping Snort:**
- /etc/init.d/snortd start # Start Snort
- /etc/init.d/snortd stop # Stop Snort
- /etc/init.d/snortd restart # Restart Snort
  - By default, Snort does not start automatically at boot (can be automated).
  - Installed version lacks database support (logs only to files).

Installation from Source Code

- **Download:**
  - Get the latest version from www.snort.org (e.g., snort-1.9.0.tar.gz).
  - Save in /opt directory.
- **Requirement:**
  - Linux: libpcap

- Windows: WinPcap (from http://winpcap.polito.it/)
- **Unpacking:**
- tar zxvf snort-1.9.0.tar.gz
- Creates /opt/snort-1.9.0 directory with subfolders like:
  - contrib – Extra utilities (e.g., ACID, MySQL scripts).
  - doc – Documentation.
  - etc – Configuration files.
  - rules – Predefined rule files.
  - src – Source code.
  - templates – For plugin developers.
- **Compiling & Installing:**

1. Run ./configure (set parameters, select components like SNMP/MySQL/SMB).
2. Run make (compiles the code).
3. Run make install (installs Snort).

- Use ./configure –help to list available options before compiling.

Other options are used to enable the following components of Snort:

• Support of MySQL database.

• Support of SNMP traps.

• Support of SMB alerts. SMB alerts are used to send pop-up windows to Microsoft Windows machines.

• Enable support of flex response. Flex response is used to terminate network sessions in real time.

After running the configure script, you can run the following two commands

to compile and install Snort files.

make

make install

The first command may take some time to complete depending upon how powerful your machine is. When you run the second command, files are installed in the appropriate directories. The make install command installs Snort binaries in /opt/snort directory as you selected --prefix=/opt/snort on the command line for the configure script.

Useful command line parameters that can be used with the `configure` script are shown in Table 2-1

Table 2-1 Command line parameters used with configure scripts

Parameter	Description
`--with-mysql`	Build support of MySQL with Snort.
`--with-snmp`	Build support of SNMP while compiling Snort. You have to use –with-openssl if you use this option.
`--with-openssl`	Enable OpenSSL support. You may need to use this when you use SNMP option.
`--with-oracle`	Enable support for Oracle database.
`--with-odbc`	Build support for ODBC in Snort.
`--enable-flexresp`	Enables use of Flex Response which allows canceling hostile connections. This is still experimental (see README.FLEXRESP file in Snort distribution).
`--enable-smbalerts`	Enable SMB alerts. Be careful using this as this invokes smbclient user space process every time it sends an alert.
`--prefix=DIR`	Set directory for installing Snort files.

You can also run the "make check" command before running the "make install" command to make sure that Snort is built properly.

After installing, run Snort to see if the executable file is working. Using the above mentioned procedure, Snort binary is installed in the /opt/snort/bin directory.

The following command just displays the basic help message of the newly built snort and command line options.

```
[root@conformix snort]# /opt/snort/bin/snort -?
Initializing Output Plugins!

-*> Snort! <*-
Version 1.9.0 (Build 209)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
USAGE: /opt/snort/bin/snort [-options] <filter options>
Options:
        -A          Set alert mode: fast, full, console,
                    or none  (alert file alerts only)
                    "unsock" enables UNIX socket logging
                    (experimental).
        -a          Display ARP packets
        -b          Log packets in tcpdump format (much
                    faster!)
        -c <rules> Use Rules File <rules>
        -C          Print out payloads with character data
                    only (no hex)
        -d          Dump the Application Layer
        -D          Run Snort in background (daemon) mode
        -e          Display the second layer header info
        -f          Turn off fflush() calls after binary log
                    writes
        -F <bpf>    Read BPF filters from file <bpf>
        -g <gname> Run snort gid as <gname> group (or gid)
                    after initialization
        -G <mode>   Add reference ids back into alert msgs
                   (modes: basic, url)
        -h <hn>     Home network = <hn>
        -i <if>     Listen on interface <if>
        -I          Add Interface name to alert output
        -l <ld>     Log to directory <ld>
        -m <umask> Set umask = <umask>
        -M <wrkst> Sends SMB message to workstations in file
                    <wrkst>
```

- **After Installation:**
  - Create default log and config directories:
  - mkdir -p /var/log/snort
  - mkdir -p /opt/snort/{etc,rules}
  - Copy default config and rule files:

- cp /opt/snort-1.9.0/etc/snort.conf /opt/snort/etc
- cp /opt/snort-1.9.0/etc/classification.config /opt/snort/etc
- cp /opt/snort-1.9.0/etc/reference.config /opt/snort/etc
- cp /opt/snort-1.9.0/rules/* /opt/snort/rules
- Adjust RULE_PATH in snort.conf if directory structure changes.

- **Run Snort:**
- /opt/snort/bin/snort -c /opt/snort/etc/snort.conf
  - Foreground mode: shows live alerts
  - Background mode: add -D option

Testing Snort

- Use ping or simple network traffic to trigger alerts.
- View alerts in:
  - Console (if started with -A console)
  - /var/log/snort/alert file

Common Errors & Fixes

- Cannot write to log directory:
  - Create /var/log/snort with proper permissions.
- Configuration file not found:
  - Use full path with -c option or copy snort.conf to home as .snortrc.

You can terminate the Snort session any time by pressing the Ctrl and C keys simultaneously.

```
===================================================
Snort analyzed 65 out of 65 packets, dropping 0(0.000%)
packets

Breakdown by protocol:                  Action Stats:
    TCP: 55          (84.615%)          ALERTS: 10
    UDP: 10          (15.385%)          LOGGED: 10
   ICMP: 0           (0.000%)           PASSED: 0
    ARP: 0           (0.000%)
  EAPOL: 0           (0.000%)
   IPv6: 0           (0.000%)
    IPX: 0           (0.000%)
  OTHER: 0           (0.000%)
DISCARD: 0           (0.000%)
===================================================
Wireless Stats:
Breakdown by type:
    Management Packets: 0          (0.000%)
    Control Packets:    0          (0.000%)
    Data Packets:       0          (0.000%)
===================================================
Fragmentation Stats:
Fragmented IP Packets: 0          (0.000%)
    Fragment Trackers: 0
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
===================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 55         (84.615%)
         Stream Trackers: 1
          Stream flushes: 0
           Segments used: 0
   Stream4 Memory Faults: 0
===================================================
Snort received signal 2, exiting
[root@conformix snort]#
```

Snort in the foreground mode

Generating Test Alerts

The following script name is snort-test.sh uses the same command as mentioned above but is useful when Snort is running in the daemon mode.

```
#!/bin/sh
#
#####################################################################
# You are free to copy and distribute this script under       #
# GNU Public License until this part is not removed           #
# from the script.                                            #
#####################################################################
#                        HOW TO USE                           #
#                                                             #
# Right after installation of Snort, run this script.        #
# It will generate alerts in /var/log/snort/alert file similar#
# to the following:                                           #
#                                                             #
# Note that Snort must be running at the time you run this    #
# script.                                                     #
#                                                             #
# [**] [1:498:3] ATTACK RESPONSES id check returned root [**] #
# [Classification: Potentially Bad Traffic] [Priority: 2]     #
# 08/31-15:56:48.188882 255.255.255.255 -> 192.168.1.111      #
# ICMP TTL:150 TOS:0x0 ID:0 IpLen:20 DgmLen:84                #
# Type:0  Code:0  ID:45596  Seq:1024  ECHO REPLY               #
#                                                             #
# These alerts are displayed at the end of the script.        #
#####################################################################
#
clear
echo "#####################################################################"
echo "#            Script to test Snort Installation                    #"
echo "#                       Written By                                #"
echo "#                                                                 #"
echo "#                    Rafeeq Rehman                                #"
echo "#                  rr@argusnetsec.com                             #"
echo "#           Argus Network Security Services Inc.                  #"
echo "#              http://www.argusnetsec.com                         #"
echo "#####################################################################"
echo

```

This script generates alerts which you can see in the /var/log/snort/ alert file (if running in daemon mode) or on the screen where Snort is running. Alerts are generated by sending ICMP echo packets with a predefined pattern in the data part.

The echo command is used for this purpose. This pattern triggers the following Snort rule, generating an alert.

alert ip any any -> any any (msg:"ATTACK RESPONSES id check

returned root"; content: "uid=0(root)"; classtype:bad-unknown;

sid:498; rev:3;)

After generating alerts, the script will display the last eighteen lines of the /var/ log/snort/alert file.

Generating Test Alerts with Automatic Snort Startup

If you installed Snort in the /opt/snort directory, you can also use the following script that will start and stop Snort by itself and verify that it is working properly.

Make sure that Snort is NOT already running before starting this script because the script starts Snort by itself. This script is found as snort-test-auto.sh file on the website http://authors.phptr.com/rehman/.

```
#!/bin/sh
#
#####################################################################
# You are free to copy and distribute this script under             #
# GNU Public License until this part is not removed                 #
# from the script.                                                  #
#####################################################################
#                          HOW TO USE                               #
#                                                                   #
# Right after installation of Snort, run this script.              #
# It is assumed that snort executable is present in the             #
# /opt/argus/bin directory and all rules and configuration          #
# files are present under /opt/argus/etc/snort directory.           #
# If files are in other locations, edit the following location      #
# of variables. If you used the installation script provided        #
# along with this script, the files will be automatically           #
# located in appropriate directories.                               #
#                                                                   #
# Note that the script starts and stops Snort by itself and         #
# you should make sure that Snort is not running at the time        #
# you run this script.                                              #
#                                                                   #
# It will generate alerts in /tmp/alert file similar                #
# to the following:                                                 #
#                                                                   #
# [**] [1:498:3] ATTACK RESPONSES id check returned root [**]       #
# [Classification: Potentially Bad Traffic] [Priority: 2]           #
# 08/31-15:56:48.188882 255.255.255.255 -> 192.168.1.111            #
# ICMP TTL:150 TOS:0x0 ID:0 IpLen:20 DgmLen:84                      #
# Type:0  Code:0  ID:45596  Seq:1024  ECHO REPLY                     #
#                                                                   #
# These alerts are displayed at the end of the script.              #
#####################################################################
#

PREFIX=/opt/snort
SNORT=$PREFIX/bin/snort
SNORT_CONFIG=$PREFIX/etc/snort.conf
LOG_DIR=/tmp
ALERT_FILE=$LOG_DIR/alert
```

Running Snort on a Non-Default Interface

On Linux systems, Snort starts listening to network traffic on Ethernet interface eth0. Many people run Snort on multi-

interface machines. If you want Snort to listen to some other interface, you have to specify it on the command line using the -i option.

The following command starts Snort so that it listens to network interface eth1.

snort -c /opt/snort/etc/snort.conf –i eth1

In case of automatic startup and shutdown as explained in the next section, you have to modify /etc/init.d/snortd script so that Snort starts on the desired interface at boot time.

Automatic Startup and Shutdown

You can configure Snort to start at boot time automatically and stop when the system shuts down. On UNIX-type machines, this can be done through a script that starts and stops Snort. The script is usually created in the /etc/init.d directory on Linux. A link to the startup script may be created in /etc/rc3.d directory and shutdown links may be present in /etc/rc2.d, /etc/rc1.d and /etc/rc0.d directories. A typical script file /etc/init.d/snortd that is bundled with Snort RPM

4. 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 eth0 and eth1 on a Linux machine.

/opt/snort/bin/snort -c /opt/snort/etc/snort.conf -i eth0 -l /

var/log/snort0

/opt/snort/bin/snort -c /opt/snort/etc/snort.conf -i eth1 -l /

var/log/snort1

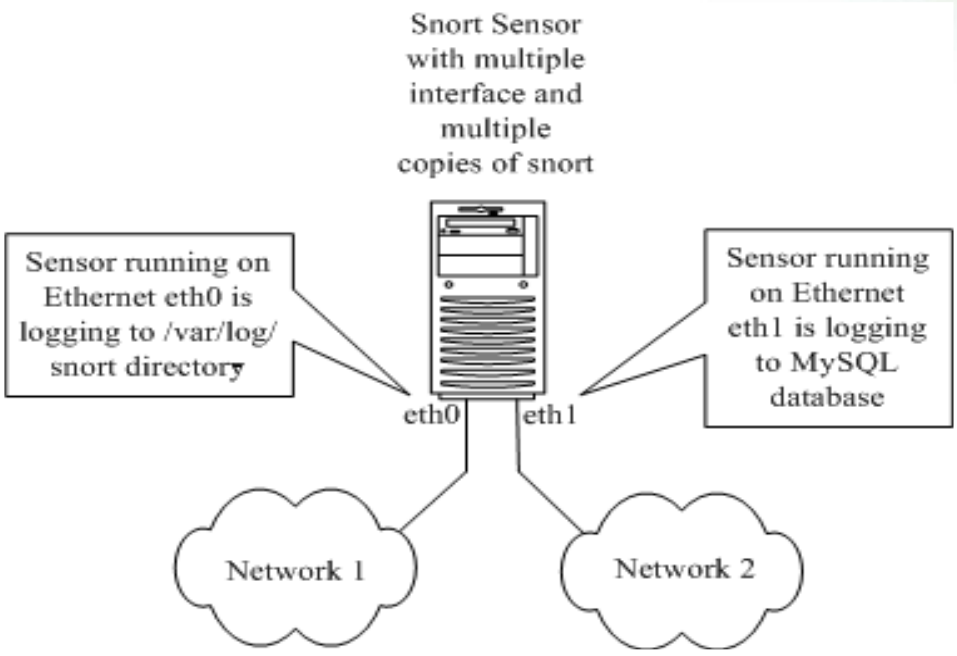
Note that you have created two log directories, /var/log/snort0 and /var/ log/snort1, so that both of the Snort sessions keep their log files separate. These directories must exist before you start Snort.

If both sessions log to a MySQL database, which is configured through

snort.conf file, the same database can be used.

Note that you can also have different configuration files for these two sessions.

There may be many reasons for having separate configuration files. The main reason is that HOME_NETWORK is different for the two sessions. Another reason may be that you want to log alert data in log files for one interface and in a database for the second interface.



Running Snort on multiple network interfaces and logging to different places.

5. SNORT COMMAND LINE OPTIONS

Table 2-2 Snort command line options

Options	Description
-A	This options sets alert mode. Alert modes are used to set different levels of detail with the alert data. Options available are fast, full, console or none. You have already seen that the console mode is used to display alert data on the console screen instead of logging to files. The fast mode is useful for high-speed operations of Snort.
-b	This option is used to log packets in `tcpdump` format. Logging is very fast and you can use the `tcpdump` program later on to display the data.
-c	This is the most commonly used option. You specify the location of `snort.conf` file with this option. When specified, Snort does not look into default locations of the configuration file snort.conf. As an example, if the `snort.conf` file is present in `/etc` directory, you will use "`-c /etc/snort.conf`" on the command line while starting Snort.
-D	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.
-i	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.
-l	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.
-M	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. Note that you can achieve the same goal through `snort.conf` file as well, which will be explained later.
-T	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.

6. STEP-BY-STEP PROCEDURE TO COMPILE AND INSTALL SNORT FROM SOURCE CODE

RPM (easy way):
rpm -i <snort_file_name.rpm>

From source (step-by-step):

1. Download from snort.org.
2. Unpack: tar zxvf <snort-x.y.z.tar.gz> → cd snort-x.y.z
3. Configure (example):
   ./configure --prefix=/opt/snort --with-mysql --with-snmp --with-openssl
4. Build & install: make && sudo make install
5. Create dirs:
   sudo mkdir -p /var/log/snort /opt/snort/{etc,rules}
6. Copy configs:
   - snort.conf → /opt/snort/etc/
   - classification.config, reference.config → /opt/snort/etc/
7. Copy rules → /opt/snort/rules/
8. Create startup script snortd in /etc/init.d and add runlevel links (/etc/rcx.d/) so Snort starts at boot.
9. If using MySQL, start MySQL before starting Snort.

Minimal run example (after setup):
snort -c /opt/snort/etc/snort.conf -D -l /var/log/snort

Location of Snort Files

- Binary:
  - RPM: /usr/sbin/snort
  - Source: set by --prefix (e.g., /opt/snort/bin/snort)
- Configs:
  - RPM: /etc/snort/snort.conf

- Source example: /opt/snort/etc/
- classification.config, reference.config live alongside snort.conf (path set inside snort.conf)

- Rules:
  - Referenced from snort.conf
  - RPM: typically /etc/snort/
  - Source example: /opt/snort/rules/
- Logs:
  - Default: /var/log/snort/ (create if missing)
  - Change via snort.conf: config logdir: /snortlog
  - Or CLI: snort -l /snortlog ...

7. SNORT MODES

Snort operates in two basic modes: **packet sniffer mode and NIDS 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.

Network Sniffer Mode:

· Works like tcpdump, captures and logs packets without intrusion detection.

· Logs can be viewed using Snort or tcpdump.

· 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
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

- Displays packet details until Ctrl+C is pressed.
- Typical packet info includes:

- 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

- To see more details (incl. application data):

snort -dv

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+.-

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

To display all packet information on the console, use the following command.

This command displays captured data in hexadecimal as well as ASCII format.

[root@conformix snort]# /opt/snort/bin/snort -dev

Logging Snort Data

1. Text Format

- Log and display packets:
- snort -dev -l /var/log/snort
- Logs stored in /var/log/snort/ as host-based directories.
    - Files named by protocol (e.g., TCP:2489-23, ICMP_ECHO).
- Data format is same as console output.

2. Binary Format (Quick Mode)

- Suitable for high-speed networks (less overhead).
- Logs all packets in tcpdump raw binary format:
- snort -l /tmp -b
- Creates file like: snort.log.1037840339 (timestamp-based).
- Each Snort run creates a new file.

3. Viewing Logged Data

- View binary log with Snort:
- snort -dev -r /tmp/snort.log.1037840339
    - Add tcp, icmp, or udp to filter specific protocols.
- View with tcpdump:
- tcpdump -r /tmp/snort.log.1037840514

Network Intrusion Detection (NIDS) Mode

- 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.

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.

Database Logging

- Snort can log alerts into a database (e.g., MySQL) for analysis

8. SNORT ALERT MODES

When running in Network Intrusion Detection (NIDS) mode, Snort generates alerts when a packet matches a rule. Alerts can be configured via command-line options or snort.conf.

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.

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.

3. UNIX Socket Mode

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

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

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

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

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.

Requirements:

- 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.

Scenarios:

1. Single network adapter (no IP assigned).
2. Dual network adapters:
   - eth1: Connected to a private isolated network (with IP).
   - eth0: Connected to the public network (no IP assigned, runs in stealth mode).

Steps:

1. Bring up the interface without an IP:
2. ifconfig eth0 up
3. Start Snort on that interface:
4. snort -c /opt/snort/etc/snort.conf -i eth0 -D

- -D runs Snort as a daemon (background mode).

9. WORKING WITH SNORT RULES

Snort Rules and Intruder Signatures

- 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).
- Snort rules can:
  1. Generate alerts

2. Log events
3. Pass packets silently (drop without alerting—different from firewall pass/drop 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:

1. Physical Layer
   - Physical media, network interface adapter, drivers.
2. Data Link Layer (Network Interface Layer)
   - Manages Ethernet (MAC) addresses.
3. Network Layer (IP Layer)
   - Uses IP protocol (RFC 791).
   - Includes ICMP protocol (RFC 792).
   - Handles point-to-point communication.
4. Transport Layer (TCP/UDP)
   - TCP (RFC 793): Reliable, connection-oriented.
   - UDP (RFC 768): Unreliable, connectionless.
5. Application Layer
   - User-facing protocols: Telnet, HTTP, FTP, etc.

The First Bad Rule

This is considered the simplest (and worst) Snort rule, but it helps verify if Snort is working:

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.

Use case:
Only for initial testing after Snort installation to ensure alerts are being generated.

Explanation of the Rule Parts

- 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.

CIDR Notation in Snort

- Snort uses CIDR (Classless Inter-Domain Routing) for specifying network addresses.
- Format: IP_address/netmask_bits
  - Example: 192.168.1.0/24 → Netmask = 255.255.255.0

- Example: /32 → A single host (e.g., 192.168.2.113/32)

Example Rule (specific host & TTL check):

alert icmp any any -> 192.168.1.113/32 any \

(msg: "Ping with TTL=100"; ttl:100;)

Triggers only for ICMP packets with TTL=100 to host 192.168.1.113.

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*.

Figure 3-1 Basic structure of Snort rules.

A Snort rule has two main parts: the rule header and the 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).**

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.).

Key Points

- A rule may detect:
  - A single intrusion type or
  - Multiple intrusion types.
- Intelligent rules are designed to match multiple intrusion signatures effectively.

Action	Protocol	Address	Port	Direction	Address	Port

Figure 3-2 Structure of Snort rule header.

10. SNORT RULE HEADER AND OPTIONS

A Snort rule is divided into two main parts: **Rule Header** and **Rule Options**.

1. Rule Header

The rule header contains the **criteria for matching packets** and **action to be taken**. Its components, in order, are:

1. **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.
2. **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.
3. **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.
4. **Direction**
   - Specifies packet flow: -> (left to right), <- (right to left), <> (either direction).
   - Determines which address/port is considered source or destination.
5. **Destination Address & Port**
   - Defines the target of the packet.
   - Can use any for all addresses/ports.

Example Rule Header:

alert icmp any any -> any any

2. 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.

Key Notes:

- **Options allow precise targeting of packet features (IP, TCP/UDP, ICMP headers, payloads).**
- **Alerts are generated only if both header conditions and options criteria are satisfied.**

Example rule:

alert icmp any any -> any any (msg: "Ping with TTL=100"; ttl: 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

Component	Value	Meaning
		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

The action in a Snort rule defines what happens when all rule conditions are met. Only when a packet meets all criteria is the action executed.

Snort has predefined actions and also allows user-defined actions.

1. Predefined Actions

1. **Pass**
   - Ignores the packet.
   - Useful to skip known safe traffic or internal vulnerability assessment hosts, improving Snort performance.
2. **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.

3. **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.
4. **Activate**
   - Generates an alert and **activates another rule** for further testing.
   - Useful when more complex, multi-step packet inspection is needed.
5. **Dynamic**
   - Can only be triggered by another rule using the activate action.
   - Not applied to packets directly.

2. User-Defined Actions

User-defined actions allow more advanced functionality:

- **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
}
```

Example: SMB alert and MySQL logging

```
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

- 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.
- Options in the rule can reference fields from other protocols.
  - Example: ttl is part of IP header but can be checked in an ICMP rule.

Example Rule:

alert icmp any any -> any any (msg: "Ping with TTL=100"; ttl: 100;)

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)

Examples:

Address	Meaning

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;)

- Address Lists: Multiple addresses can be excluded in brackets, separated by commas.

  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 Field in Snort Rules

The direction field determines which addresses and ports in the rule are treated as source and destination.

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	alert tcp any any

Symbol	Meaning	Example Use
	= source, Left side = destination	<- 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.

- The direction field is critical for determining which side of the connection the rule applies to.

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

Rule Options

Options come after the rule header inside **( )**.

- Separated by **;** (semicolon).
- **All options must be true** for the rule to trigger.
- Options = keyword + argument.
  - Example: msg:"TCP ping detected"

Important Keywords

☑ ack

- Matches TCP **Acknowledgement Number**.
- Used to detect **TCP ping scans**.
- alert tcp any any -> 192.168.1.0/24 any (flags:A; ack:0; msg:"TCP ping detected";)

☑ classtype

- Groups rules by **type + priority**.
- Defined in classification.config.
- Example:
- classtype:DoS; priority:1
- ⚠ **Lower number = higher priority!**

The following rule uses

default priority with the classification DoS:

alert udp any any -> 192.168.1.0/24 6838 (msg:"DoS"; \

content: "server"; classtype:DoS;)

The following is the same rule but we override the default priority used for the

classification.

alert udp any any -> 192.168.1.0/24 6838 (msg:"DoS"; \

content: "server"; classtype:DoS; priority:1)

Using classifications and priorities for rules and alerts, you can distinguish

between high- and low-risk alerts. This feature is very useful when you want to escalate high-risk alerts or want to pay attention to them first.

☑ content

- Searches for a **string or pattern** inside packet data.
- Example:
- alert tcp any any -> any any (content:"GET"; msg:"GET found";)
- Can use **ASCII** or **HEX** (|47 45 54| = GET).
- Extra helpers:
  - **offset** → where to start search.

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";)

You can use the depth keyword to define the point after which Snort should stop

searching the pattern in the data packets.

  - **depth** → where to stop search.

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";)

This keyword is very important since you can use it to limit searching inside the

packet. For example, information about HTTP GET requests is found in the start of the

packet.

- **nocase** → ignore upper/lower case.

☑ content-list

- Reads search patterns from a file.
- Example file porn contains:
- porn
- hardcore
- under 18
- Rule:
- alert ip any any -> 192.168.1.0/24 any (content-list:"porn"; msg:"Porn word matched";)

☑ dsize

- Checks **data size** in a packet.
- Detects **buffer overflow attempts**.
- alert ip any any -> any any (dsize:>6000; msg:"Large packet";)

☑ flags

- Matches **TCP header flags**.
- Examples:
- S = SYN

- F = FIN
- A = ACK

- Can combine with ! (NOT), + (AND), * (OR).
- alert tcp any any -> 192.168.1.0/24 any (flags:SF; msg:"SYN-FIN scan detected";)

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

☑ fragbits

- Matches IP fragmentation bits.
- D = Don't Fragment, M = More Fragments, R = Reserved.
- alert icmp any any -> any any (fragbits:D; msg:"DF set";)

☑ icmp_id / icmp_seq

- Matches **ID** or **Sequence number** in ICMP packets.
- Used in ping/Echo Request-Reply detection.

The general format for using this keyword

is as follows:

icmp_id: <ICMP_id_number>

An ICMP identified field is found in ICMP ECHO REQUEST and ICMP ECHO REPLY messages as discussed in RFC 792. This field is used to match ECHO REQUEST and ECHO REPLY messages. Usually when you use the ping command, both of these types of ICMP packets are exchanged between sending and receiving hosts. The sending host sends ECHO REQUEST packets and the destination host replies with ECHO REPLY-type ICMP packets. This field is useful for discovering which packet is the reply to a particular request. The following rule checks if the ICMP ID field in the ICMP header is equal to 100. It generates an alert if this criterion is met.

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";)

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

- ☑ **id**

- Matches **IP packet fragment ID**.
- Useful for detecting **special attacks using fixed IDs**.

- ☑ **ipopts**

- Matches **IP header options** (like Record Route-rr, Timestamp-ts, etc.).

- Common options:
  - rr → Record Route
  - ts → Timestamps
  - lsrr → Loose Source Routing
  - ssrr → Strict Source Routing

- Example:
- alert ip any any -> any any (ipopts: lsrr; msg:"Loose source routing attempt";)

☑ ip_proto

- Detects the **protocol number** in the IP header.
- Uses either **name** or **number**.
- Example:
- alert ip any any -> any any (ip_proto: 94; msg:"IPIP tunneling";)

☑ logto

- Logs packets to a specific file in /var/log/snort.
- Example:
- alert icmp any any -> any any (logto:mylog; ttl:100;)

☑ msg

- Adds a **message** to logs/alerts.
- Example: msg:"TCP SYN Scan";

☑ nocase

- Used with **content** keyword.
- Makes search **case-insensitive**.

☑ priority

- Assigns **priority number** to alerts.
- **1 = highest priority**.
- Example:
- priority:1;

☑ react

- Used to **block or warn** sessions (requires special compilation).
- Example:
- alert tcp 192.168.1.0/24 any -> any 80 (msg:"Block HTTP"; react:block;)

☑ reference

- Adds a **link to external info** (like CVE or Bugtraq).
- Example:
- reference:cve,CAN-2001-0876;

☑resp (Flexible Response)

- Sends a **response packet** to disrupt hacker activity.
- Options:
  - rst_snd → Reset to sender
  - rst_rcv → Reset to receiver
  - rst_all → Reset both sides
  - icmp_net → ICMP network unreachable
- Example:
- alert tcp any any -> 192.168.1.0/24 8080 (resp:rst_snd;)

☑rev

- Shows the **revision number** of a rule.
- Example: rev:2;

☑rpc

- Detects **RPC requests**.
- Takes: Application number, Procedure, Version.
- Example:

- alert ip any any -> any any (rpc:10000,*,3; msg:"RPC request";)

☑ sameip

- Detects if **source and destination IPs are the same** (spoofing attempt).
- Example: sameip;

☑ seq

- Tests **TCP sequence number**.
- Example: seq:1000;

☑ flow

- Matches **TCP session direction/state**.
- Options:
  - to_client, to_server, from_client, from_server
  - established, stateless, stream_only, no_stream

☑ session

- Dumps **TCP session data**.
- Example:
- log tcp any any -> 192.168.1.0/24 110 (session:printable;)

☑ sid (Snort ID)

- Gives each rule a unique ID.
- Range:
  - 0–99 reserved
  - 100–1,000,000 Snort rules
  - >1,000,000 local rules
- Example: sid:1000001;

☑ tag

- Logs **extra packets** after a rule triggers.
- Syntax: tag:type,count,metric[,direction]
  - type: session | host
  - metric: packets | seconds
- Example:
- tag:session,100,packets;

☑ tos

- Detects a specific **Type of Service (TOS)** value.
- Example: tos:1

☑ ttl

- Detects **Time To Live (TTL)** in IP header.
- Example: ttl:100;
- Can detect **traceroute attempts**.

☑ uricontent

- Like **content**, but searches only the **URI part** of HTTP requests.

11. 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

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

◇ 1. 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.

◇ 2. Variables as Lists

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

◇ 3. Variables with Interface Names

- Variables can be based on interface IPs.

- Example (Linux):
- var HOME_NET $eth0_ADDRESS
- var EXTERNAL_NET $eth1_ADDRESS
- Benefit: If interface IPs change → rules adapt automatically.

    - 4. Using any

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

var EXTERNAL_NET any

The config Directives

The config directives in the snort.conf file allow a user to configure many general settings for Snort. Examples include the location of log files, the order of applying rules and so on. These directives can be used to replace many command line options as well. The general format of applying a config directive is as follows:

config directive_name[: value]

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`.

Table 3-6 Snort config directives (continued)

Directive	Description
umask	Equivalent to –m command line option. Using this option you can set the UMASK while running Snort.
pkt_count	Equivalent to –n command line option. Using this directive you can exit from Snort after a defined number of packets.
nolog	Equivalent to –N command line option. Logging is disabled except alerts. Remember, alerts are really both alerts and logs.
obfuscate	Equivalent to –O command line option. It is used to obfuscate IP addresses so that you are able to send the logs for analysis to someone without disclosing the identity of your network.
no_promisc	Equivalent to –p command line option and is used to disable promiscuous mode.
quiet	Equivalent to –q command line option. This will disable banner information at Snort startup time and prevent statistical information from being displayed.
chroot	Equivalent to –t command line option. It is used to change root directory for Snort to a specific directory.
checksum_mode	Used to checksum for particular types of packets. It takes arguments such as none, noip, notcp, noicmp, noudp, and all.
set_uid	Equivalent to –u command line option and is used to set user ID for the Snort process.
utc	Equivalent to –U command line option and is used to use UTC instead of local time in alerts and logs.
verbose	Equivalent to –v command line option. It is used to log messages to standard output in addition to standard logging.
dump_payload_verbose	Equivalent to –X command line option. This dumps the received raw packet on the standard output.
show_year	Equivalent to –y command line option and is used to display year in the timestamp.
stateful	Used to set assurance mode for stream4 preprocessor. Preprocessors are discussed in detail in Chapter 4.

You have already seen how the classification directive is used in the classification. config file. As another example, the following line is used to start Snort in the daemon mode.

config daemon

You can also use –D command line option to start Snort in the daemon mode.

Preprocessor Configuration

- Preprocessors = input plug-ins → process packets *before* Snort rules.
- Syntax:
- preprocessor <name>[: options]
- Examples:
  - preprocessor frag2 → handles IP defragmentation.
  - preprocessor stream4: detect_scans → detects port scans.
- You can also write custom preprocessors if needed.

◇ Output Module Configuration

- Output modules = output plug-ins → decide where & how alerts/logs are stored.
- Syntax:
- output <module_name>[: options]
- Example: log to MySQL database:
- output database: alert, mysql, user=rr password=boota dbname=snort host=localhost
- Output modules can also send logs to SNMP traps, files, or other systems.

◇ Defining New Action Types

- Rules start with an action (e.g., alert, log, pass).
- You can define custom action types in snort.conf.

- Example: dump_database action logs to both database + tcpdump file.
- ruletype dump_database {
- type alert
- output database: alert, mysql, user=rr dbname=snort host=localhost
- output log_tcpdump: tcpdump_log_file
- }
-
- dump_database icmp any any -> 192.168.1.0/24 any (fragbits:D; msg:"DF bit set";)
- Benefit → one rule = multiple outputs.

◇ Rules Configuration

- Last part of snort.conf.
- You can write rules directly here, but usually rules are kept in separate files (*.rules).
- Example: local rules can be placed in local.rules.

◇ Include Files

- Use include keyword to add rule files into main config.
- Syntax:
- include myrules.rules
- Most default Snort rule files (*.rules) are included this way.
- Not required to end with .rules, but it's the convention.

Order of Rules Based on Action

- Snort has 5 types of rules (alert, log, pass, activate, dynamic).
- They fall into 3 categories checked in this order (default):
  1. Alert rules

2. Pass rules
3. Log rules

- Alternative order (faster but risky):
  1. Pass → 2. Alert → 3. Log
  - Use with caution (-o command line switch or config order in snort.conf).
  - One bad pass rule can let attacks slip through!
- Custom rule types are checked last.
  - Example: Alert -> Pass -> Log -> snmp_alerts.

◇ Automatically Updating Snort Rules

Keeping Snort rules up to date = essential. Two methods:

☑ Simple Method (Shell Script)

- Uses wget to download rules archive.
- Steps:
  1. Download from Snort website.
  2. Extract rules (tar -zxf).
  3. Backup old rules.
  4. Move new rules into /etc/snort.
  5. Restart Snort (/etc/init.d/snortd restart).

☑ Sophisticated Method (Oinkmaster)

- 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.

◇ Sample Default 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).

◇ Writing Good Rules

Best practices for strong Snort rules:

1. Always include a msg (description).
2. Use classtype to categorize rule.
3. Assign a unique SID.
4. Add reference (CVE, bugtraq, etc.).
5. Use rev to track versions.
6. Write generalized rules that detect variations of attacks.

12. PREPROCESSORS AND OUTPUT MODULES

◇ Preprocessors

- 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.

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

3. frag2

- Defragments IP packets.
- Default: 4MB memory, 60 sec timeout.
- Example:
- preprocessor frag2: 2097152, 30

4. stream4

- Functions:
  1. TCP stream reassembly
  2. Stateful inspection
- Modules: stream4 & stream4_reassemble.
- Example options: timeout, memcap, ports (21, 23, 25, 53, 80, etc.).

5. SPADE (Statistical Packet Anomaly Detection Engine)

- Detects unusual/anomalous packets.
- Needs memory & processing power.

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

◇ Output Modules

- Control how Snort sends logs/alerts.
- Configured in snort.conf with keyword output.

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).

Examples:

- 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 BPF (Berkeley Packet Filters)

- Filters packets at data link level before Snort rules.
- Saves CPU by only analyzing relevant traffic.
- Example:
- ip[1] != 0

snort -F bpf.txt -c /opt/snort/etc/snort.conf

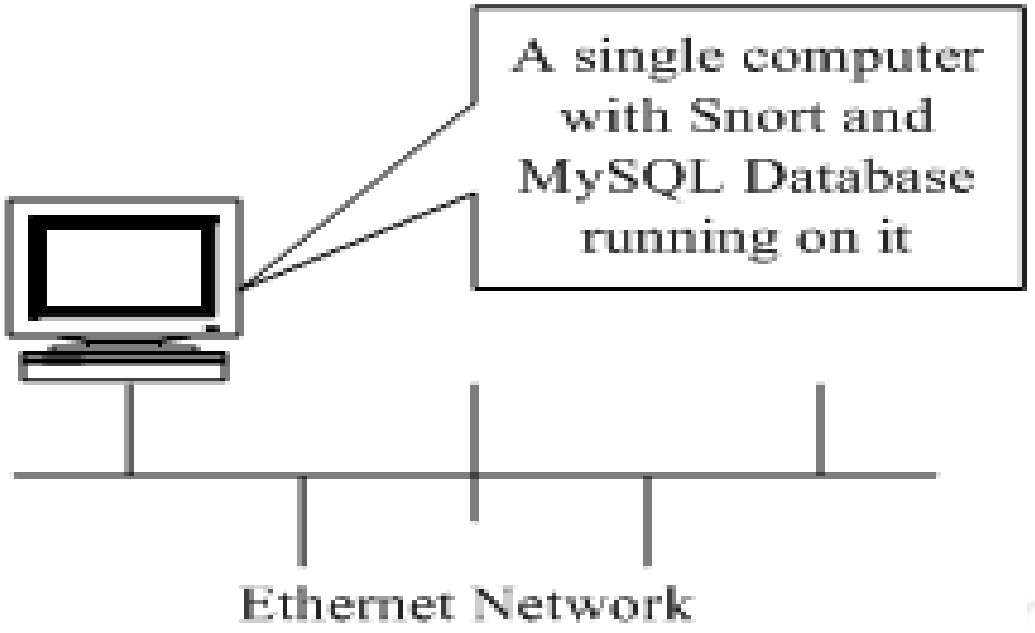
13. aUSING SNORT WITH MYSQL

Snort can log alerts in different formats such as text files or tcpdump files, but using a database like MySQL provides more powerful features. By storing alerts in a database, it becomes easier to maintain historical data, generate detailed reports, and analyze attack trends using external tools like ACID (Analysis Console for Intrusion Databases). MySQL is free, reliable, cross-platform, and therefore commonly used as the backend for Snort.

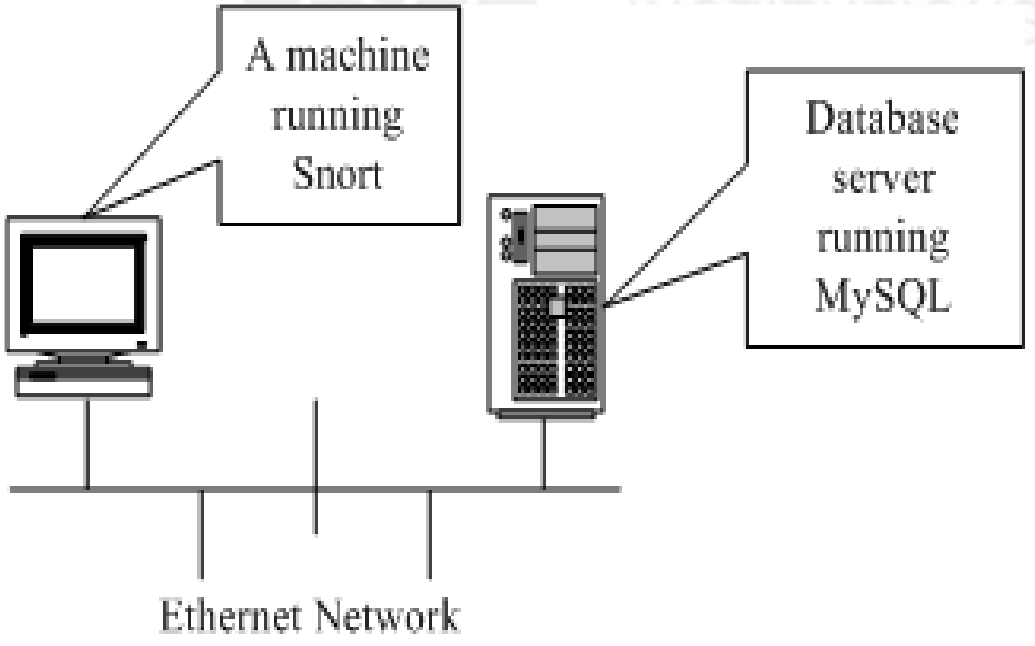
Some different scenarios for using a database with Snort are:

• You can install and run the MySQL database server on the same machine where

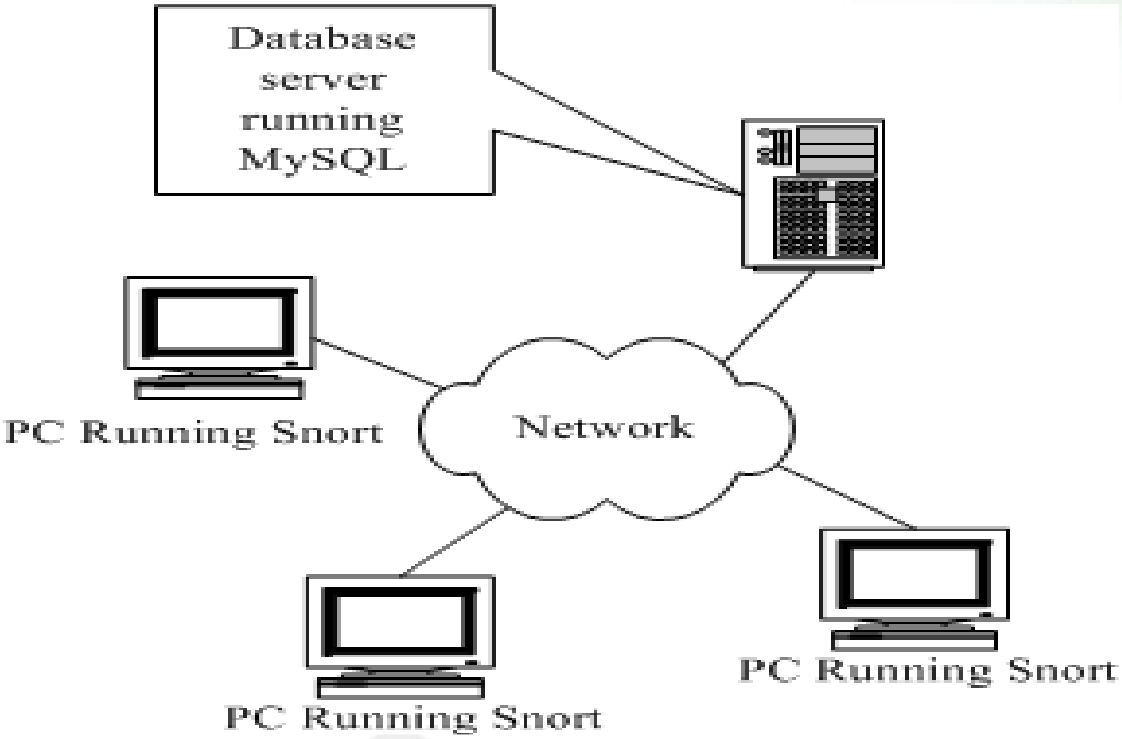
Snort is running, as shown in Figure below.



A single computer running Snort and MySQL database server.



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



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

How to Use MySQL with Snort

Snort can be configured with MySQL in multiple deployment models:

- 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.

If Snort logs are sent across the network, there is a risk that attackers could intercept or tamper with the information. To prevent this, administrators can use Stunnel, which encrypts the communication between Snort

sensors and the database server, ensuring that sensitive IDS logs remain protected.

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 following commands are used in this session:

• 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 isdisplayed after login and you get the "mysql>" prompt where you can issue other commands.

Making Snort Work with MySQL

• The command "create database snort;" creates a new database in the MySQL server with the name "snort". You can use any name of your choice for the database.

• The "use snort" command is used to start using the newly created database.

• The "status" command shows current status of the database server. It shows that the currently opened database is "snort."

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.

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

```
[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.

Database Maintenance

Because Snort generates a very large volume of alerts, the MySQL database can grow quickly and become inefficient if not maintained. Regular database maintenance is essential.

One important command is OPTIMIZE TABLE, which defragments and reorganizes the table to improve performance:

optimize table data;

This process can be automated using scheduled tasks (cron jobs) to run daily or weekly.

For long-term management, administrators can choose to archive old data. Two approaches are common:

1. Back up the database, drop it, and recreate it from scratch.
2. Move old records to archive tables, keeping the active database smaller and faster.

Database maintenance ensures that Snort remains efficient and capable of handling large amounts of data without performance issues.

Lecture Slides

Lecture Slides
https://drive.google.com/drive/folders/1MdIU8YLJpG_n3byErvlaSDCn4JTDGb23?usp=drive_link

Lab exercises

1.Enhance Snort's functionality using preprocessors and plugins.

2.Set up advanced alerting and logging mechanisms.

Assignment

Assignment

1. Configure Snort in packet logger mode and demonstrate how logs are stored in a specific directory. Submit screenshots of configuration and sample log output.
2. Write and test three custom Snort rules:
   - One to detect HTTP traffic,
   - One to detect ICMP echo requests,
   - One to detect suspicious FTP access.
3. Set up Snort with a MySQL backend and log alerts into the database. Verify entries using SQL queries and explain the purpose of at least three important Snort tables.
4. Demonstrate the use of variables in snort.conf by creating a HOME_NET and EXTERNAL_NET, then write rules using those variables. Show how changing variables affects rule application.
5. Configure and test Stunnel with Snort and MySQL to securely transmit alerts from a sensor to a central server. Submit configuration details and results of a sample attack logged securely.

Part A Q & A

PART -A

1. What is Snort? (CO3, K1)

Answer: Snort is an open-source Intrusion Detection System (IDS) that monitors network traffic in real-time. It can detect suspicious or malicious activity and act as a sniffer, logger, or intrusion prevention system depending on the configuration.

2. List the main modes of Snort operation. (CO3, K1)

Answer: Snort works in three major modes: Sniffer mode (captures packets), Packet logger mode (saves packets to files), and NIDS mode (analyzes traffic against rules). These modes provide flexibility for administrators.

3. What is the function of Snort rules? (CO3, K2)

Answer: Snort rules define what traffic to monitor and what action to take. They describe conditions like protocols, IPs, or ports and actions like alerting or logging. Without rules, Snort cannot detect attacks.

4. Define a preprocessor in Snort. (CO3, K2)

Answer: A preprocessor is a plugin that processes packets before rules are applied. It can normalize, reassemble, or detect anomalies in packets. Examples include frag2 (defragmentation) and stream4 (TCP reassembly).

5. What are Snort output modules? (CO3, K2)

Answer: Output modules define where Snort logs alerts. They can log to plain text, syslog, MySQL, or XML/CSV formats. They allow administrators to analyze events easily.

6. Write the general structure of a Snort rule. (CO3, K2)

Answer: A Snort rule has a header and options. The header specifies action, protocol, source/destination, and direction, while the options define conditions like payload content. Example:

alert tcp any any -> 192.168.1.0/24 80 (msg:"HTTP detected";)

7. What is the role of the keyword msg in Snort rules? (CO3, K1)

Answer: The msg keyword specifies a message to display when a rule is triggered. It helps administrators quickly understand what the alert represents in the logs.

8. Explain the use of the content keyword. (CO3, K2)

Answer: The content keyword searches for specific strings or data patterns inside packet payloads. It is used to detect known malicious commands or exploits embedded in traffic.

9. What is the purpose of the flags keyword in Snort? (CO3, K2)

Answer: The flags keyword checks TCP flag settings like SYN, FIN, or ACK. This helps detect unusual scans such as Xmas or NULL scans, which are used in reconnaissance.

10. Differentiate between -> and <> in Snort rules. (CO3, K2)

Answer: The -> operator checks traffic in one direction, from source to destination. The <> operator checks both directions, ensuring bidirectional monitoring.

11. What is the significance of classtype in Snort rules? (CO3, K2)

Answer: classtype assigns rules into categories of attacks like DoS, web-attack, or attempted-admin. This helps organize alerts and understand their severity.

12. What is the use of sid in Snort rules? (CO3, K1)

Answer: sid stands for Snort ID and uniquely identifies each rule. It prevents confusion between rules and makes it easy to manage updates or references.

13. Mention the importance of rev in Snort rules. (CO3, K1)

Answer: The rev keyword shows the revision number of a rule. It indicates whether a rule is updated or changed, helping administrators keep rules up-to-date.

14. What are the advantages of using variables in Snort configuration? (CO3, K2)

Answer: Variables make rules reusable and environment-friendly. By using $HOME_NET or $EXTERNAL_NET, one can adapt rules without editing IP addresses everywhere.

15. Explain the function of the dsize keyword. (CO3, K2)

Answer: The dsize keyword checks payload size of packets. Large or abnormal sizes often indicate buffer overflow attacks or other malicious activity.

16. What is the role of the itype and icode keywords? (CO3, K2)

Answer: itype matches the ICMP type field and icode matches the ICMP code field. They allow detection of specific ICMP messages such as Echo Requests or Redirects.

17. Explain frag2 preprocessor. (CO3, K2)

Answer: The frag2 preprocessor performs IP packet defragmentation. Attackers may use fragmented packets to evade detection, and frag2 reassembles them for inspection.

18. Explain stream4 preprocessor. (CO3, K2)

Answer: The stream4 preprocessor reassembles TCP streams into complete sessions. This allows Snort to detect attacks spread across multiple packets, improving accuracy.

19. What is portscan detection in Snort? (CO3, K2)

Answer: Portscan preprocessor detects multiple connection attempts across ports or hosts. Since scanning often precedes intrusions, detecting it gives early warning of attacks.

20. What is the difference between alert and log actions in Snort? (CO3, K2)

Answer: alert both logs and generates a message, while log only records the packet details. Alerts notify administrators, whereas logs provide passive monitoring.

21. Define Oinkmaster. (CO3, K2)

Answer: Oinkmaster is a Perl script for updating Snort rules automatically. It downloads new rules, removes unwanted ones, and ensures rules remain current with minimal effort.

22. What is the use of the include keyword in snort.conf? (CO3, K1)

Answer: The include keyword allows adding external rule files into the main configuration. This keeps configuration neat and makes managing multiple rule sets easier.

23. What are the categories of default Snort rule files? (CO3, K1)

Answer: Snort rules are grouped into categories like backdoor, dos, dns, telnet, web-attacks, icmp, smtp etc. This categorization makes it easier to manage and apply rules.

24. How is database output used in Snort? (CO3, K3)

Answer: Snort can log events directly into MySQL or PostgreSQL databases using an output plugin. This allows advanced querying and integration with reporting tools like ACID.

25. What is the role of Stunnel in Snort logging? (CO3, K3)

Answer: Stunnel encrypts Snort's communication with remote MySQL databases. It prevents attackers from intercepting or altering IDS logs sent over untrusted networks.

26. Differentiate between alert_fast and alert_full output plugins. (CO3, K2)

Answer: alert_fast logs one-line alerts, suitable for high-speed traffic, while alert_full logs detailed multi-line alerts, giving more context but with slower performance.

27. What is the purpose of Berkeley Packet Filters BPF) in Snort? (CO3, K3)

Answer: BPF allows filtering packets at the kernel level before Snort processes them. It saves CPU cycles by ignoring irrelevant traffic and focusing only on useful packets.

28. Explain the use of nocase in Snort rules. (CO3, K2)

Answer: The nocase keyword makes content searches case-insensitive. It ensures that even if attackers change text case (e.g., "GET" vs "get"), Snort still detects the match.

29. What is SPADE in Snort? (CO3, K2)

Answer: SPADE is the Statistical Packet Anomaly Detection Engine. It detects unusual traffic patterns using statistics, catching attacks that may not match predefined rules.

30. Why are variables like HOME_NET and EXTERNAL_NET important? (CO3, K2)

Answer: These variables define the internal and external networks in Snort. By changing them once, rules automatically adapt to different network environments.

31. What is the role of revision numbers in Snort rules? (CO3, K1)

Answer: Revision numbers (rev) indicate the version of a rule. When rules are updated or modified, the revision number is increased for better tracking.

32. What is the significance of unified output in Snort? (CO3, K3)

Answer: Unified output is a binary format for logging alerts. It is efficient and works with tools like Barnyard to convert logs later into other formats for analysis.

33. Explain the difference between pass and drop actions. (CO3, K3)

Answer: pass ignores packets and allows them through, while drop blocks packets (when Snort runs inline) and also generates an alert. Drop is used for prevention, not just detection.

34. What is the role of rule references in Snort? (CO3, K2)

Answer: Rule references link Snort rules to external vulnerability databases like CVE or Bugtraq. This helps administrators understand what vulnerability or exploit the rule corresponds to.

35. Why is database maintenance important in Snort? (CO3, K3)

Answer: Snort generates large amounts of alert data, which can slow down the database. Regular optimization and archiving keep the database efficient, ensuring fast queries and stable storage.

Part B Q

PART -B

1. Explain the architecture of Snort in detail with a neat diagram. (CO3, K2)
2. Write short notes on the different modes of Snort operation with examples. (CO3, K2)
3. Describe the general structure of Snort rules with suitable examples for each component. (CO3, K2)
4. Discuss the purpose and working of preprocessors in Snort. Explain with examples like frag2 and stream4. (CO3, K2)
5. Explain the role of variables in Snort configuration. Show how HOME_NET and EXTERNAL_NET are used in rules. (CO3, K3)
6. Illustrate the output modules in Snort and explain how alerts can be logged to files, syslog, and databases. (CO3, K3)
7. Describe the steps in configuring Snort with MySQL. Include compilation, database setup, user creation, and snort.conf configuration. (CO3, K3)
8. Explain in detail the different keywords used in Snort rules such as msg, content, flags, dsize, and classtype. (CO3, K2)
9. Write short notes on Snort rule management tools such as Oinkmaster and discuss their significance. (CO3, K3)
10. Discuss the importance of database maintenance in Snort with MySQL and explain methods like OPTIMIZE TABLE and archiving. (CO3, K4)

Supportive Online Certification courses

R.M.K GROUP OF INSTITUTIONS

SUPPORTIVE ONLINE COURSES

S No	Course provider	Course title	Link
1	Udemy	The Complete Cyber Security Course : Network Security	https://www.udemy.com/course/network-security-course/?couponCode=ST12MT90625AI
2	Udemy	Snort Intrusion Detection, Rule Writing, and PCAP Analysis	https://www.udemy.com/course/snort-intrusion-detection-rule-writing-and-pcap-analysis/?couponCode=ST12MT90625AI
3	NPTEL	Network Security	https://onlinecourses.nptel.ac.in/noc25_ee54/preview

Real life Applications in day to day life and to Industry

REAL TIME APPLICATIONS IN DAY TO DAY LIFE AND TO INDUSTRY

1. Enterprise Network Security

Snort acts as an IDS/IPS in organizations, detecting attacks like malware, DoS, or port scans in real time. It helps protect sensitive company data and ensures smooth business operations.

2. Home/Personal Network Monitoring

Individuals use Snort to secure their home Wi-Fi by detecting unknown devices and malicious traffic. Parents can also monitor or block unsafe websites for children.

3. Banking and Financial Services

Banks deploy Snort to monitor online transactions and detect fraud or intrusion attempts. This ensures secure financial services and builds customer trust.

Content beyond Syllabus

Contents beyond the Syllabus

1. Suricata: https://suricata.io/

Assessment Schedule

FIAT: 14.08.2025

SIAT: 23.09.2025

Model Exam: 28.10.2025

R.M.K GROUP OF INSTITUTIONS

Prescribed Text books & Reference books

R.M.K GROUP OF INSTITUTIONS

PRESCRIBED TEXT BOOKS AND REFERENCE BOOKS

TEXT BOOKS

1. Rafeeq Rehman : “Intrusion Detection with SNORT, Apache, MySQL, PHP and ACID,” 1st Edition, Prentice Hall , 2003.

2. Internet Security: A Hands-on Approach, by Wenliang Du, Third Edition, 2019

REFERENCE BOOKS

1. Christopher Kruegel, Fredrik Valeur, Giovanni Vigna: “Intrusion Detection and Correlation Challenges and Solutions”, 1st Edition, Springer, 2005.

2. Carl Endorf, Eugene Schultz and Jim Mellander “Intrusion Detection & Prevention”, 1st Edition, Tata McGraw-Hill, 2004.

3. Stephen Northcutt, Judy Novak : “Network Intrusion Detection”, 3rd Edition, New Riders Publishing, 2002.

4. T. Fahringer, R. Prodan, “A Text book on Grid Application Development and Computing Environment”. 6th Edition, Khanna Publishers, 2012.

Mini Project Suggestions

R.M.K GROUP OF INSTITUTIONS

MINI PROJECT SUGGESTIONS

1. Snort-based Home Network Intrusion Detection (CO3, K3)

Deploy Snort on a Raspberry Pi or Linux PC connected to a home Wi-Fi router.

Capture and analyze malicious traffic (e.g., port scans, ping sweeps).

Outcome: Demonstrate alerts for real-world home traffic threats.

2. Custom Snort Rule Development and Testing (CO3, K3)

Write a set of custom Snort rules for detecting:

- Unauthorized SSH login attempts
- HTTP traffic with suspicious keywords
- Large ICMP ping packets (DoS simulation)

Outcome: Show how rules trigger alerts with sample attack simulations.

3. Snort + MySQL + Web Dashboard (CO3, K4)

Configure Snort to log alerts into a MySQL database.

Build a simple PHP or Python Flask web app to display alerts in a dashboard.

Outcome: A mini Security Information and Event Management (SIEM) system.

4. Snort vs Suricata: Performance Comparison (CO3, K5)

Install Snort and Suricata on the same network testbed.

Run simulated traffic (normal + attack) and compare detection speed, CPU usage, and false positives.

Outcome: Research-style comparative study with charts and conclusions.

5. Snort with Stunnel for Secure Log Transmission (CO3, K4)

Set up Snort on one machine to log alerts.

Use Stunnel to encrypt logs and send them securely to a central MySQL server.

Outcome: Demonstrate secure IDS log collection for distributed networks.

Thank you