**1.CRC**

import java.util.Scanner;

import java.io.\*;

public class crc {

    public static void main(String args[]) {

       Scanner sc = new Scanner(System.in);

       //Input Data Stream

       System.out.print("Enter message bits: ");

       String message = sc.nextLine();

       System.out.print("Enter generator: ");

       String generator = sc.nextLine();

       int data[] = new int[message.length() + generator.length() - 1];

       int divisor[] = new int[generator.length()];

       for(int i=0;i<message.length();i++){

          data[i] = Integer.parseInt(message.charAt(i)+"");

       }

       for(int i=0;i<generator.length();i++){

          divisor[i] = Integer.parseInt(generator.charAt(i)+"");

       }

       //Calculation of CRC

       for(int i=0;i<message.length();i++){

            if(data[i]==1){

                for(int j=0;j<divisor.length;j++){

                   data[i+j] ^= divisor[j];

                }

            }

       }

       //Display CRC

       System.out.print("The checksum code is: ");

       for(int i=0;i<message.length();i++){

           data[i] = Integer.parseInt(message.charAt(i)+"");

       }

       for(int i=0;i<data.length;i++) {

           System.out.print(data[i]);

       }

       System.out.println();

       //Check for input CRC code

       System.out.print("Enter checksum code: ");

       message = sc.nextLine();

       System.out.print("Enter generator: ");

       generator = sc.nextLine();

       data = new int[message.length() + generator.length() - 1];

       divisor = new int[generator.length()];

       for(int i=0;i<message.length();i++){

           data[i] = Integer.parseInt(message.charAt(i)+"");

       }

       for(int i=0;i<generator.length();i++){

           divisor[i] = Integer.parseInt(generator.charAt(i)+"");

       }

       //Calculation of remainder

       for(int i=0;i<message.length();i++) {

           if(data[i]==1){

            for(int j=0;j<divisor.length;j++){

              data[i+j] ^= divisor[j];

               }

           }

       }

       //Display validity of data

       boolean valid = true;

       for(int i=0;i<data.length;i++){

          if(data[i]==1){

          valid = false;

          break;

          }

       }

       if(valid==true) {

           System.out.println("Data stream is valid");

       }

       else {

           System.out.println("Data stream is invalid. CRC error occurred.");

       }

   }

}

**2.DIJKSTRA**

class dijks{

    static final int V=9;

    public static void main(String args[]){

        int bfs[][] = new int[][] { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },

                            { 4, 0, 8, 0, 0, 0, 0, 11, 0 },

                            { 0, 8, 0, 7, 0, 4, 0, 0, 2 },

                            { 0, 0, 7, 0, 9, 14, 0, 0, 0 },

                            { 0, 0, 0, 9, 0, 10, 0, 0, 0 },

                            { 0, 0, 4, 14, 10, 0, 2, 0, 0 },

                            { 0, 0, 0, 0, 0, 2, 0, 1, 6 },

                            { 8, 11, 0, 0, 0, 0, 1, 0, 7 },

                            { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

        dijks t = new dijks();

        t.Dijkstra(bfs,0);

    }

    public static void Dijkstra(int bfs[][],int src){

        int d[] = new int[V];

        boolean set[] = new boolean[V];

        for(int i=0;i<V;i++){

            d[i] = Integer.MAX\_VALUE;

            set[i] = false;

        }

        d[src] = 0;

        for(int c=0;c<V-1;c++){

            int u = minDis(d,set);

            set[u] = true;

            for(int v=0;v<V;v++){

                if(!set[v] && d[u]!=Integer.MAX\_VALUE && bfs[u][v]!=0 && d[u]+bfs[u][v]<d[v]){

                    d[v] = d[u]+bfs[u][v];

                }

            }

        }

        printSol(d);

    }

    public static int minDis(int d[],boolean set[]){

        int min = Integer.MAX\_VALUE;

        int min\_index = -1;

        for(int i=0;i<V;i++){

            if(set[i]==false && d[i]<=min){

                min = d[i];

                min\_index = i;

            }

        }

        return min\_index;

    }

    public static void printSol(int d[]){

        System.out.println("Vertex\t\tdist from src");

        for(int i=0;i<V;i++){

            System.out.println(i+"\t\t"+d[i]);

        }

    }

}

**3.DISTANCE VECTOR ROUTING**

import java.io.\*;

public class DVR {

    static int graph[][];

    static int via[][];

    static int rt[][];

    static int v;

    static int e;

    public static void main(String args[]) throws IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Please enter the number of Vertices: ");

        v = Integer.parseInt(br.readLine());

        System.out.println("Please enter the number of Edges: ");

        e = Integer.parseInt(br.readLine());

        graph = new int[v][v];

        via = new int[v][v];

        rt = new int[v][v];

        for (int i = 0; i < v; i++)

            for (int j = 0; j < v; j++) {

                if (i == j)

                    graph[i][j] = 0;

                else

                    graph[i][j] = 9999;

            }

        for (int i = 0; i < e; i++) {

            System.out.println("Please enter data for Edge " + (i + 1) + ":");

            System.out.print("Source: ");

            int s = Integer.parseInt(br.readLine());

            s--;

            System.out.print("Destination: ");

            int d = Integer.parseInt(br.readLine());

            d--;

            System.out.print("Cost: ");

            int c = Integer.parseInt(br.readLine());

            graph[s][d] = c;

            graph[d][s] = c;

        }

        dvr\_calc\_disp("The initial Routing Tables are: ");

        System.out.print("Please enter the Source Node for the edge whose cost has changed: ");

        int s = Integer.parseInt(br.readLine());

        s--;

        System.out.print("Please enter the Destination Node for the edge whose cost has changed: ");

        int d = Integer.parseInt(br.readLine());

        d--;

        System.out.print("Please enter the new cost: ");

        int c = Integer.parseInt(br.readLine());

        graph[s][d] = c;

        graph[d][s] = c;

        dvr\_calc\_disp("The new Routing Tables are: ");

    }

    static void dvr\_calc\_disp(String message) {

        System.out.println();

        init\_tables();

        update\_tables();

        System.out.println(message);

        print\_tables();

        System.out.println();

    }

    static void update\_table(int source) {

        for (int i = 0; i < v; i++) {

            if (graph[source][i] != 9999) {

                int dist = graph[source][i];

                for (int j = 0; j < v; j++) {

                    int inter\_dist = rt[i][j];

                    if (via[i][j] == source)

                        inter\_dist = 9999;

                    if (dist + inter\_dist < rt[source][j]) {

                        rt[source][j] = dist + inter\_dist;

                        via[source][j] = i;

                    }

                }

            }

        }

    }

    static void update\_tables() {

        int k = 0;

        for (int i = 0; i < 4 \* v; i++) {

            update\_table(k);

            k++;

            if (k == v)

                k = 0;

        }

    }

    static void init\_tables() {

        for (int i = 0; i < v; i++) {

            for (int j = 0; j < v; j++) {

                if (i == j) {

                    rt[i][j] = 0;

                    via[i][j] = i;

                } else {

                    rt[i][j] = 9999;

                    via[i][j] = 100;

                }

            }

        }

    }

    static void print\_tables() {

        for (int i = 0; i < v; i++) {

            for (int j = 0; j < v; j++) {

                System.out.print("Dist: " + rt[i][j] + "    ");

            }

            System.out.println();

        }

    }

}

**4.LEAKY BUCKET**

import java.util.Scanner;

import java.lang.\*;

public class Leaky {

  public static void main(String[] args) {

    int i;

    int a[] = new int[20];

    int buck\_rem = 0, buck\_cap = 4, rate = 3, sent, recv;

    Scanner in = new Scanner(System.in);

    System.out.println("Enter the number of packets");

    int n = in.nextInt();

    System.out.println("Enter the packets");

    for (i = 1; i <= n; i++)

      a[i] = in.nextInt();

    System.out.println("Clock \t packet size \t accept \t sent \t remaining");

    for (i = 1; i <= n; i++) {

      if (a[i] != 0) {

        if (buck\_rem + a[i] > buck\_cap)

          recv = -1;

        else {

          recv = a[i];

          buck\_rem += a[i];

        }

      } else

        recv = 0;

      if (buck\_rem != 0) {

        if (buck\_rem < rate) {

          sent = buck\_rem;

          buck\_rem = 0;

        } else {

          sent = rate;

          buck\_rem = buck\_rem - rate;

        }

      } else

        sent = 0;

      if (recv == -1)

        System.out.println(+i + "\t\t" + a[i] + "\t dropped \t" + sent + "\t" + buck\_rem);

      else

        System.out.println(+i + "\t\t" + a[i] + "\t\t" + recv + "\t" + sent + "\t" + buck\_rem);

    }

  }

}

**5.TOKEN BUCKET**

import java.util.Scanner;

public class Token {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter token bucket capacity: ");

        int tokenBucketCapacity = scanner.nextInt();

        System.out.print("Enter token rate (tokens per second): ");

        int tokenRate = scanner.nextInt();

        System.out.print("Enter packet size: ");

        int packetSize = scanner.nextInt();

        System.out.print("Enter the number of packets to send: ");

        int numPackets = scanner.nextInt();

        int tokens = 0;

        for (int i = 0; i < numPackets; i++) {

            tokens += tokenRate;

            if (tokens >= packetSize) {

                System.out.printf("Sending packet %d (Size: %d bytes)\n", i + 1, packetSize);

                tokens -= packetSize;

            } else {

                System.out.printf("Dropping packet %d (Size: %d bytes)\n", i + 1, packetSize);

            }

        }

    }

}

**6.TCP CLIENT**

import java.net.\*;

import java.io.\*;

public class TCPClient {

    private Socket socket = null;

    private BufferedReader input = null;

    private DataOutputStream out = null;

    public TCPClient(String address, int port) {

        try {

            socket = new Socket(address, port);

            System.out.println("Connected");

            // Using try-with-resources for automatic resource management

            input = new BufferedReader(new InputStreamReader(System.in));

            out = new DataOutputStream(socket.getOutputStream());

            // Read and send user input until "Over" is entered

            String line = "";

            while (!line.equals("Over")) {

                try {

                    line = input.readLine();

                    out.writeUTF(line);

                } catch (IOException i) {

                    System.out.println(i);

                }

            }

        } catch (UnknownHostException u) {

            System.out.println(u);

        } catch (IOException i) {

            System.out.println(i);

        } finally {

            // Close resources in a finally block to ensure they are closed

            try {

                if (input != null) input.close();

                if (out != null) out.close();

                if (socket != null) socket.close();

            } catch (IOException i) {

                System.out.println(i);

            }

        }

    }

    public static void main(String args[]) {

        // Create an instance of the Client class

        TCPClient client = new TCPClient("127.0.0.1", 5000);

    }

}

**7.TCP SERVER**

import java.net.\*;

import java.io.\*;

public class TCPServer {

    //initialize socket and input stream

    private Socket socket = null;

    private ServerSocket server = null;

    private DataInputStream in = null;

    // constructor with port

    public TCPServer(int port) {

        // starts server and waits for a connection

        try {

            server = new ServerSocket(port);

            System.out.println("Server started");

            System.out.println("Waiting for a client ...");

            socket = server.accept();

            System.out.println("Client accepted");

            // takes input from the client socket

            in = new DataInputStream(new BufferedInputStream(socket.getInputStream()));

            String line = "";

            // reads message from client until "Over" is sent

            while (!line.equals("Over")) {

                try {

                    line = in .readUTF();

                    System.out.println(line);

                } catch (IOException i) {

                    System.out.println(i);

                }

            }

            System.out.println("Closing connection");

            // close connection

            socket.close(); in .close();

        } catch (IOException i) {

            System.out.println(i);

        }

    }

    public static void main(String args[]) {

        TCPServer server = new TCPServer(5000);

    }

}

**8.UDP CLIENT:**

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class UDPClient {

    public static void main(String args[]) throws IOException {

        Scanner sc = new Scanner(System.in);

        // Step 1:Create the socket object for

        // carrying the data.

        DatagramSocket ds = new DatagramSocket();

        InetAddress ip = InetAddress.getLocalHost();

        byte buf[] = null;

        // loop while user not enters "bye"

        while (true) {

            String inp = sc.nextLine();

            // convert the String input into the byte array.

            buf = inp.getBytes();

            // Step 2 : Create the datagramPacket for sending

            // the data.

            DatagramPacket DpSend =

                new DatagramPacket(buf, buf.length, ip, 1234);

            // Step 3 : invoke the send call to actually send

            // the data.

            ds.send(DpSend);

            // break the loop if user enters "bye"

            if (inp.equals("bye"))

                break;

        }

    }

}

**9.UDP SERVER:**

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.net.SocketException;

public class UDPServer {

    public static void main(String[] args) throws IOException {

        // Step 1 : Create a socket to listen at port 1234

        DatagramSocket ds = new DatagramSocket(1234);

        byte[] receive = new byte[65535];

        DatagramPacket DpReceive = null;

        while (true) {

            // Step 2 : create a DatgramPacket to receive the data.

            DpReceive = new DatagramPacket(receive, receive.length);

            // Step 3 : revieve the data in byte buffer.

            ds.receive(DpReceive);

            System.out.println("Client:-" + data(receive));

            // Exit the server if the client sends "bye"

            if (data(receive).toString().equals("bye")) {

                System.out.println("Client sent bye.....EXITING");

                break;

            }

            // Clear the buffer after every message.

            receive = new byte[65535];

        }

    }

    // A utility method to convert the byte array

    // data into a string representation.

    public static StringBuilder data(byte[] a) {

        if (a == null)

            return null;

        StringBuilder ret = new StringBuilder();

        int i = 0;

        while (a[i] != 0) {

            ret.append((char) a[i]);

            i++;

        }

        return ret;

    }

}

**Installation of rootkits and study about the variety of options**

Rootkit is a stealth type of malicious software designed to hide the existence of

certain process from normal methods of detection and enables continued privileged access to a computer.

INTRODUCTION:

Breaking the term rootkit into the two component words, root and kit, is a useful way to define it. Root is a UNIX/Linux term that's the equivalent of Administrator in Windows.

The word kit denotes programs that allow someone to obtain root/admin-level access to the computer by executing the programs in the kit — all of which is done without end-user consent or knowledge.

A rootkit is a type of malicious software that is activated each time your system boots up. Rootkits are difficult to detect because they are activated before your system's Operating System has completely booted up. A rootkit often allows the installation of hidden files, processes, hidden user accounts, and more in the systems OS. Rootkits are able to intercept data from terminals,network connections, and the keyboard.

Rootkits have two primary functions: remote command/control (back door) and

software eavesdropping. Rootkits allow someone, legitimate or otherwise, to administratively control a computer. This means executing files, accessing logs, monitoring user activity, and even changing the computer's configuration.

Therefore, in the strictest sense, even versions of VNC are rootkits. This surprises most people, as they consider rootkits to be solely malware, but in of themselves they aren't malicious at all.

The presence of a rootkit on a network was first documented in the early 1990s. At that time, Sun and Linux operating systems were the primary targets for a hacker looking to install a rootkit. Today, rootkits are available for a number of operating systems, including Windows, and are increasingly difficult to detect on any network.

**PROCEDURE:**

STEP-1: Download Rootkit Tool from GMER website www.gmer.net.

STEP-2: This displays the Processes, Modules, Services, Files, Registry, RootKit

Malwares, Autostart, CMD of local host.

STEP-3: Select Processes menu and kill any unwanted process if any.

STEP-4: Modules menu displays the various system files like .sys, .dll

STEP-5: Services menu displays the complete services running with Autostart, Enable, Disable, System, Boot.

STEP-6: Files menu displays full files on Hard-Disk volumes.

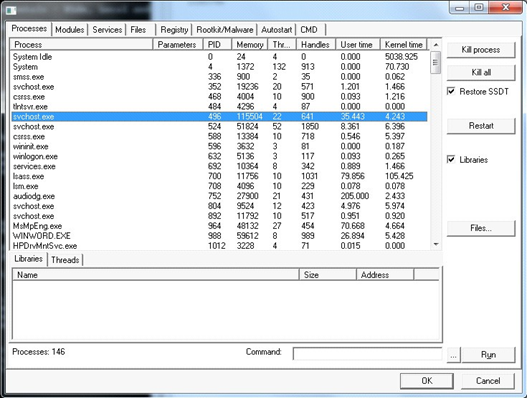
STEP-7: Registry displays Hkey\_Current\_user and Hkey\_Local\_Machine.

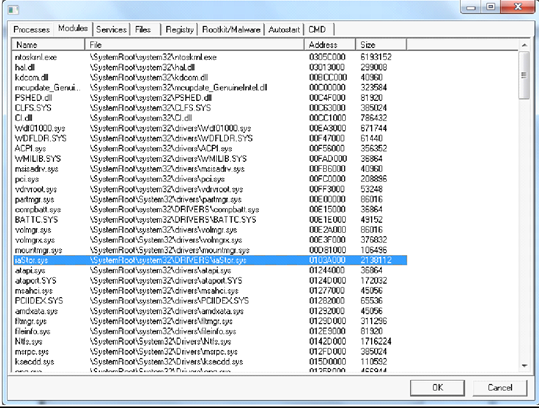
STEP-8: Rootkits / Malwares scans the local drives selected.

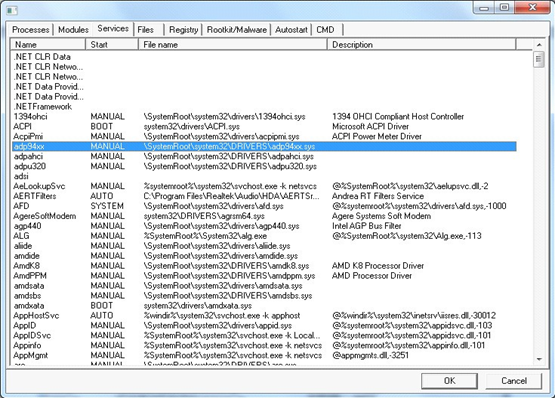
STEP-9: Autostart displays the registry base Autostart applications.

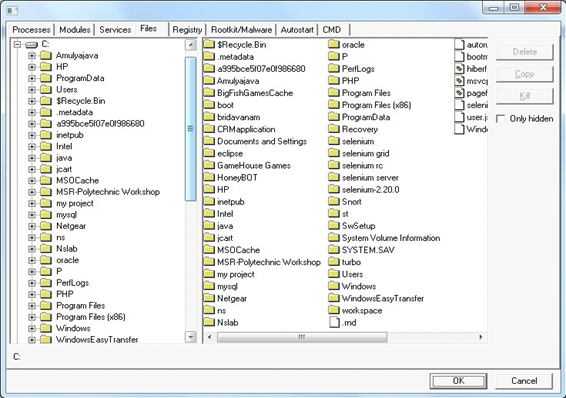
STEP-10:CMD allows the user to interact with command line utilities or Registry

**OUTPUTS**









GMER is an application that detects and removes rootkits .

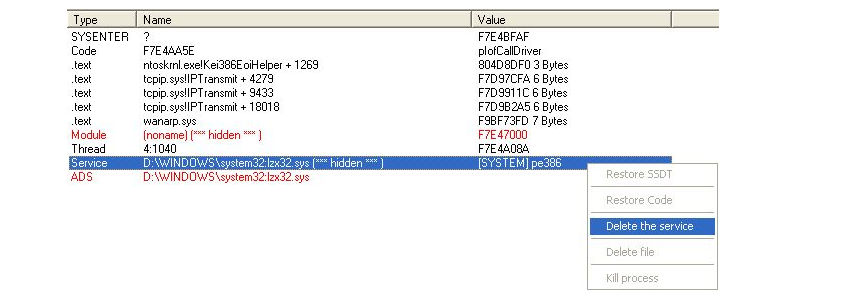
It scans for:

1. hidden processes
2. hidden threads
3. hidden modules
4. hidden services
5. hidden files
6. hidden disk sectors (MBR)
7. hidden Alternate Data Streams
8. hidden registry keys
9. drivers hooking SSDT
10. drivers hooking IDT
11. drivers hooking IRP calls
12. inline hooks

**Frequently Asked Questions**

**Frequently Asked Questions**

| ***Question:*** | ***Do I have a rootkit?*** |
| --- | --- |
| **Answer:** | **You can scan the system for rootkits using GMER. Run gmer.exe, select Rootkit tab and click the "Scan" button.**  **If you don't know how to interpret the output, please Save the log and send it to my email address.**  **Warning ! Please, do not select the "Show all" checkbox during the scan.** |
| ***Question:*** | ***How to create "3rd party" log ?*** |
| **Answer:** | **Tick "3rd party" option and then click the "Scan" button. After the scan you can use "Remove signed" and "Remove duplicates" options to filter the scan results.** |
| ***Question:*** | ***How to install the GMER software ?*** |
| **Answer:** | **Just run gmer.exe. All required files will be copied to the system during the first lanuch.** |
| ***Question:*** | ***How to uninstall/remove the GMER software from my machine ?*** |
| **Answer:** | **Just delete the exe file.** |
| ***Question:*** | ***My computer is infected and GMER won't start:*** |
| **Answer:** | **Try to rename gmer.exe to iexplore.exe and then run it.** |
| ***Question:*** | ***How do I remove the Rustock rootkit ?*** |
| **Answer:** | **When GMER detects hidden service click "Delete the service" and answer YES to all questions** |

****

| ***Question:*** | ***How do I show all NTFS Streams ?*** |
| --- | --- |
| **Answer:** | **On the "Rootkit Tab" select only: Files + ADS + Show all options and then click the Scan button.** |
| ***Question:*** | ***Can I launch GMER in Safe Mode ?*** |
| **Answer:** | **Yes, you can launch GMER in Safe Mode, however rootkits which don't work in Safe Mode won't be detected.** |
| ***Question:*** | ***I am confused as to use delete or disable the hidden "service".*** |
| **Answer:** | **Sometimes "delete the service" option wont work because the rootkit protects its service. So, in such case use: 1) "disable the service", 2) reboot your machine, and 3) "delete the service".** |

|

[Snort](https://linuxhint.com/category/snort/)

**Intrusion Detection with Snort Tutorial**

The general thought is that if a firewall is protecting one’s network, the network is considered secure. However, that is not entirely true. Firewalls are a fundamental component of a network, but they cannot fully protect the network from forced entries or hostile intent. **Intrusion Detection Systems** are used to evaluate aggressive or unexpected packets and generate an alert before these programs can harm the network. A host-based Intrusion Detection System runs on all the devices in a network or connects to an organization’s internal network. A network-based Intrusion Detection System is instead deployed at a certain point or group of points from which all ingoing and outgoing traffic can be monitored. An advantage of a host-based Intrusion Detection System is that it also can detect anomalies or malicious traffic being generated from the host itself, i.e., if the host is affected by malware, etc. **Intrusion Detection Systems (IDS)** work by monitoring and analyzing network traffic and comparing it with an established ruleset, determining what should be taken as normal for the network (i.e., for ports, bandwidths, etc.) and what to take a closer look at.

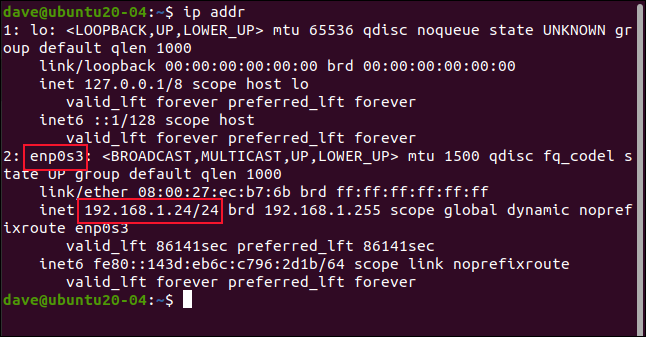
An Intrusion Detection System can be deployed depending upon the size of the network. There are dozens of quality commercial IDSs, but many companies and small businesses cannot afford them. **Snort** is a flexible, lightweight, and popular Intrusion Detection System that can be deployed according to the needs of the network, ranging from small to large networks, and provides all the features of a paid IDS. **Snort** does not cost anything but that does not mean that it cannot provide the same functionalities as an elite, commercial IDS. **Snort** is considered a passive IDS, which means it sniffs network packets, compares with the ruleset, and, in the case of detecting a malicious log or entry (i.e., detecting an intrusion), generates an alert or places an entry in a log file. **Snort** is used for monitoring the operations and activities of routers, firewalls, and servers. Snort provides a user-friendly interface, containing a chain of rulesets that can be very helpful to a person who is unfamiliar with IDSs. Snort generates an alarm in case of an intrusion (buffer overflow attacks, DNS poisoning, OS fingerprinting, port scans, and much more), giving an organization greater visibility of the network traffic and making it much easier to meet security regulations.

**1.To install Snort on Ubuntu, use this command:** sudo apt install snort

As the installation proceeds, you’ll be asked a couple of questions. You can find the answers to these by using the ip addr command before starting the installation, or in a separate terminal window.

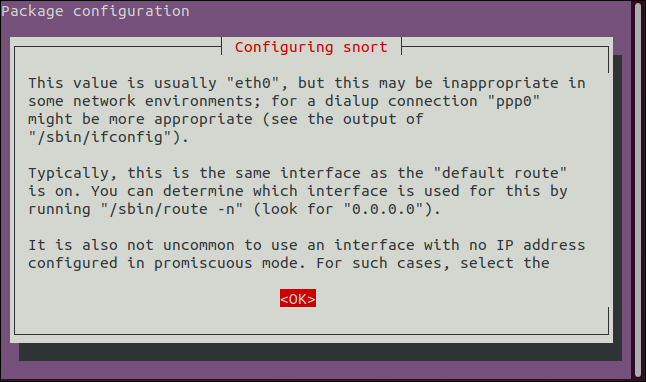
Type ip addr in separate terminal

You will get this output,it changes according to the your system

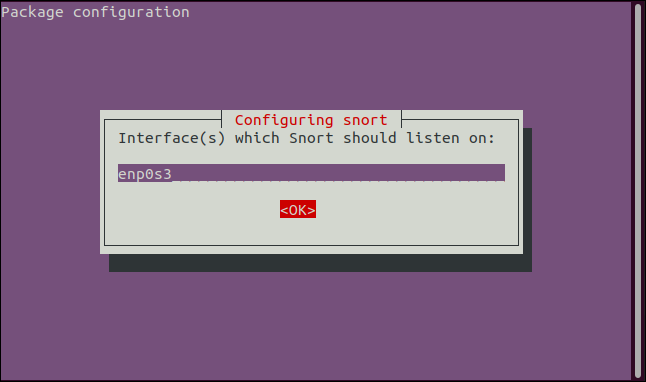


Take note of your network interface name. On this research computer, it is enp0s3

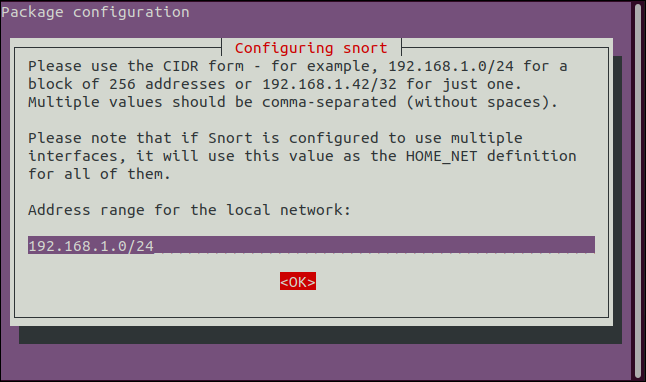
Press “Tab” to highlight the “OK” button, and press “Enter.”



Type the name of the network interface name and press “Tab” to highlight the “OK” button, and press “Enter.”



Type the network address range in CIDR format, press “Tab” to highlight the “OK” button, and press “Enter.”



To see whether snort installed or not

snort --help

You will get output

Then after that type the command **man snort**

**Output**

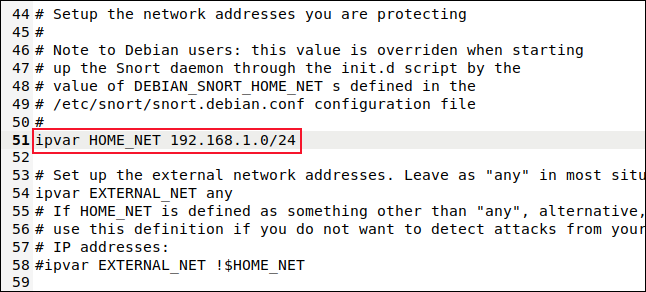
**It will give descriptions and options then after enter q**

**Configuring Snort**

**There are a few steps to complete before we can run Snort. We need to edit the “snort.conf” file.**

**sudo gedit /etc/snort/snort.conf**

**Locate the line that reads “ipvar HOME\_NET any” and edit it to replace the “any” with the CIDR notation address range of your network.**

****

**Save your changes and close the file.**

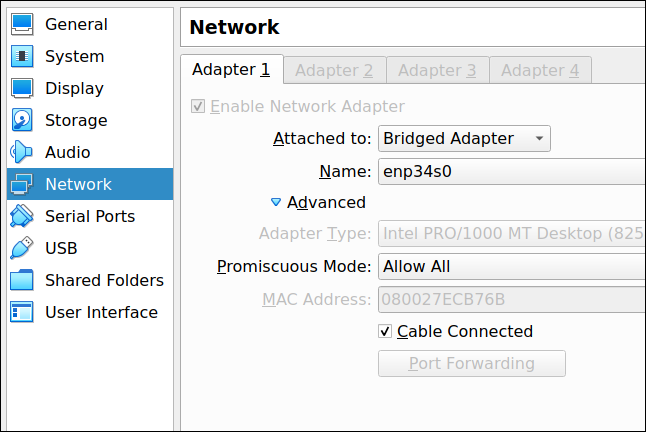
**To make the Snort computer’s network interface listen to all network traffic, we need to set it to promiscuous mode.**

**sudo ip link set enp0s3 promisc on**

**Eno1**

**sudo ip link set eno1 promisc on**

**If you are running Snort in a virtual machine, also remember to adjust the settings in your hypervisor for the virtual network card used by your virtual machine. For example, in VirtualBox, you need to go to Settings > Network > Advanced and change the “Promiscuous Mode” drop-down to “Allow All.”**

****

**sudo snort -T -i enp0s3 -c/etc/snort/snort.conf**

**sudo snort -T -i eno1 -c/etc/snort/snort.conf**

**Perform ping in command prompt by typing your inet address**

**Ping 192.168.0.167**

**Open other terminal in ubuntu and perform again ping**

**Ping 192.168.0.167**

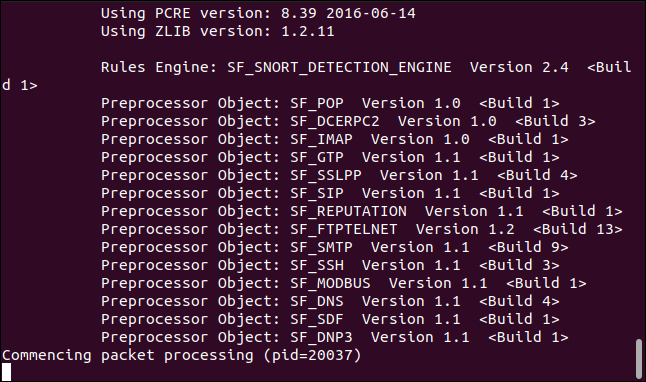
**Find icmp packets**

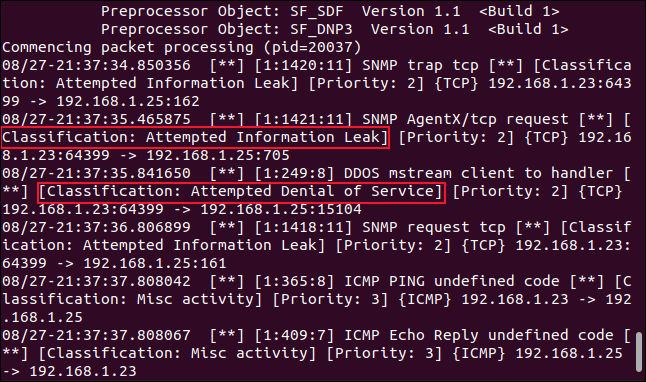
**Sudo snort -A console -i enp0s3 -c/etc/snort/snort.conf**

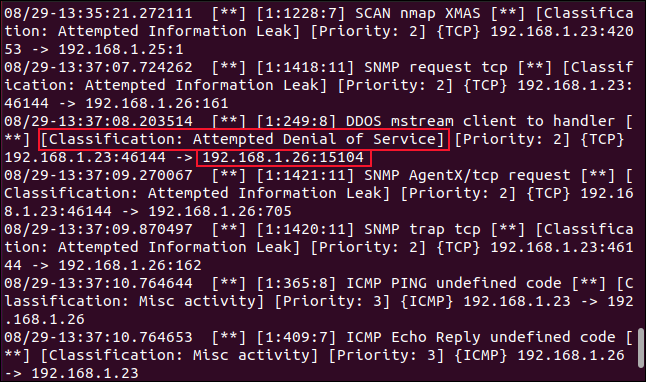
**sudo snort -A console -i enp0s3 -c/etc/snort/snort.conf**

**Eno1**

**sudo snort -A console -i eno1 -c/etc/snort/snort.conf**

****

****

****

**After that Store icmp packets in a particular file**

**sudo snort -A console -i eno1 -c/etc/snort/snort.conf**

**sudo snort -A console -i eno1 -c/etc/snort/snort.conf | tee -a santhosh.txt**

Web link for install

<http://www.keyfocus.net/kfsensor/>

Honey Pot is a device placed on Computer Network specifically designed to capture malicious network traffic. KF Sensor is the tool to setup as honeypot when KF Sensor is running it places a siren icon in the windows system tray in the bottom right of the screen. If there are no alerts then green icon is displayed.

INTRODUCTION: HONEY POT:

A honeypot is a computer system that is set up to act as a decoy to lure cyber attackers, and to detect, deflect or study attempts to gain unauthorized access to information systems. Generally, it consists of a computer, applications, and data that simulate the behavior of a real system that appears to be part of a network but is actually isolated and closely monitored. All communications with a honeypot are considered hostile, as there's no reason for legitimate users to access a honeypot. Viewing and logging this activity can provide an insight into the level and types of threat a network infrastructure faces while distracting attackers away from assets of real value.

Honeypots can be classified based on their deployment (use/action) and based on their level of involvement. Based on deployment, honeypots may be classified as:

1. Production honeypots

2. Research honeypots

Production honeypots are easy to use, capture only limited information, and are used primarily by companies or corporations. Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security. Normally, production honeypots are low-interaction honeypots, which are easier to deploy. They give less information about the attacks or attackers than research honeypots.

Research honeypots are run to gather information about the motives and tactics of the Black hat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better protect against those threats.

KF SENSOR:

KFSensor is a Windows based honeypot Intrusion Detection System (IDS). It acts as a honeypot to attract and detect hackers and worms by simulating vulnerable system services and trojans. By acting as a decoy server it can divert attacks from critical systems and provide a higher level of information than can be achieved by using firewalls and NIDS alone. KFSensor is a system installed in a network in order to divert and study an attacker’s behavior. This is a new technique that is very effective in detecting attacks.

The main feature of KFSensor is that every connection it receives is a suspect hence it results in very few false alerts. At the heart of KFSensor sits a powerful internet daemon service that is built to handle multiple ports and IP addresses. It is written to resist denial of service and buffer overflow attacks. Building on this flexibility KFSensor can respond to connections in a variety of ways, from simple port listening and basic services (such as echo), to complex simulations of standard system services. For the HTTP protocol KFSensor accurately simulates the way Microsoft’s web server (IIS) responds to both valid and invalid requests. As well as being able to host a website it also handles complexities such as range requests and client side cache negotiations. This makes it extremely difficult for an attacker to fingerprint, or identify KFSensor as a honeypot.

KFSensor installs a new system tray (systray) icon in the shape of a siren on the desktop. You click the siren icon to launch the KFSensor monitor, and it is used liberally throughout the program to indicate KFSensor’s current status. On the desktop, it will usually be gray, but it will flash red or yellow, based on event activity. The systray icon will flash until you view the KFSensor monitor, although its behavior can be customized. Low-priority events are just logged, and no alert is generated. Medium-priority events alert and make the systray icon flash yellow and orange. High-priority events alert and make the systray icon flash red and orange.

### Emulating Services with KFSensor

Each port in the Port view represents a listener. Listeners are attached to actions. Actions can be close, close and read, or call up a simulated server. The close action will immediately close the connection and log the event. Read and close will wait for the visitor to send a request, and then close the connection without sending a response. A listener can also be attached to a server action.

KFSensor calls emulated services *sim servers*, short for simulated servers. A single instance of KFSensor can have an unlimited number of sim servers defined, although only 256 can be active at once. KFSensor has two types of sim servers: *sim banner* and *sim standard*. Some services, like FTP and SMTP, exist as both sim banner servers and sim standard servers, and listen on TCP or UDP ports, depending on the requirements of the environment.

#### Sim Banner Servers

Sim banner servers are simple port listeners with the ability to serve up text or encoded data as a banner in response to a visitor request. Each sim banner server can be edited, and new banner sim servers can be added.

The default sim banner servers include Echo (7), Daytime (13), Quote of the Day (17), Chargen (19), MyDoom worm (3127), Dameware (6129), and the SubSeven trojan (54283).

#### Sim Standard Servers

Sim standard servers entail a higher level of interaction than a mere one-time banner response. KFSensor currently comes with the following emulated services:

* FTP (Guild, not Microsoft, on port 21)
* Telnet (port 23)
* SMTP (Microsoft Exchange Server 2003 on port 25)
* HTTP (IIS 6.0 and Apache on ports 80, 81, 82, and 83)
* POP3 (Exchange Server on port 110)
* NetBIOS (ports 137, 138, 139, and 445)
* SOCKS Proxy (port 1080)
* Microsoft SQL Server (ports 1433 and 1434)
* SubSeven trojan (ports 2794, 7215, and 27374)
* Hogle SMTP trojan (port 3355)
* Terminal Server (port 3389)
* HTTP Proxy (port 8080)
* VNC (port 5900)

honeypot should be set up just like the real server so that data can appear to be authentic by showing fake files, fake ports, fake directories, etc. As the honeypot creates the illusion of being legitimate; the attacker tends to believe that they have gained accessed of the real deal.

KFSensor is a honeypot for a windows system. it also acts as an IDS. Its job is to attract and detect all the attackers in the network, hence the name ‘Honeypot’. It does so by imitating a vulnerable environment and disguising itself as a server and it way, it succeeds to not only catch the attacker but also helps to know their motive

KFSensor’s role is to be a decoy server for the attackers in order to protect the real thing.

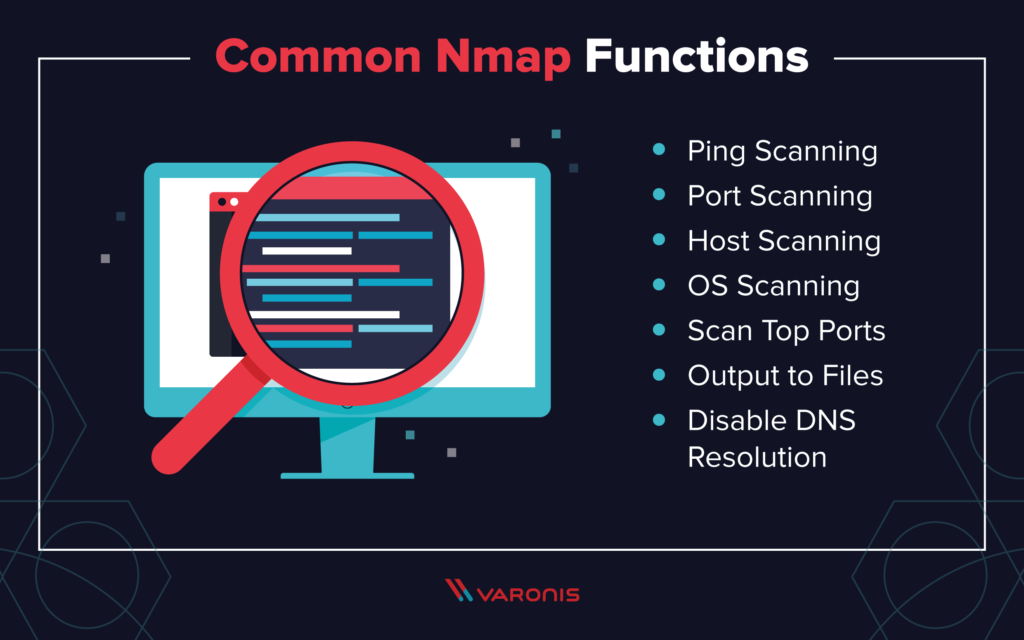
It does its job perfectly by opening fake ports on the system where it’s installed and gathering the information when a connection is made. It does this in precisely the same way as a routine server program, such as a web server or an SMTP server. By doing this it sets up a target, or a honeypot server, that will record the activities of an attacker.

**What is Nmap?**

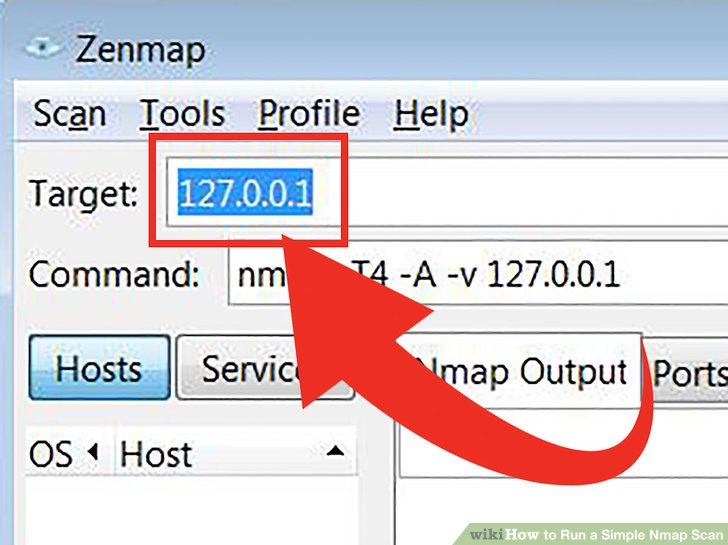
Nmap ,short for Network Mapper, is a network discovery and security auditing tool. It is known for its simple and easy to remember flags that provide powerful scanning options. Nmap is widely used by network administrators to scan for:

* Open ports and services
* Discover services along with their versions
* Guess the operating system running on a target machine
* Get accurate packet routes till the target machine
* Monitoring hosts

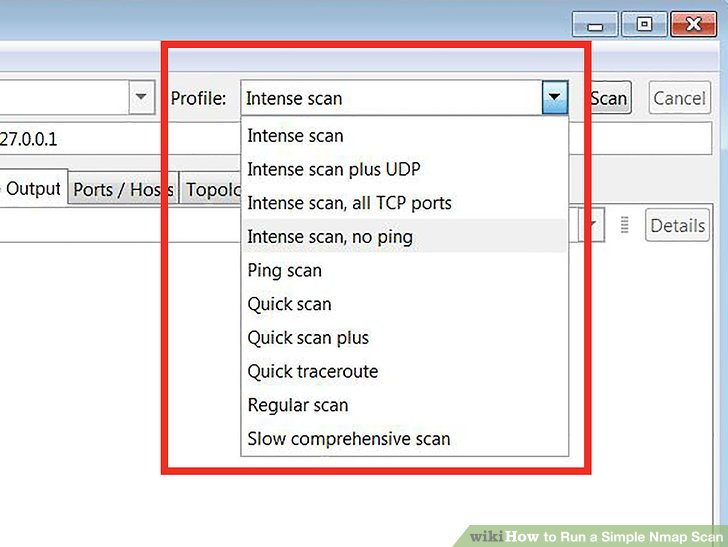
Nmap is a network mapper that has emerged as one of the most popular, free network discovery tools on the market. Nmap is now one of the core tools used by network administrators to map their networks. The program can be used to find live hosts on a network, [perform port scanning](https://www.varonis.com/blog/port-scanning-techniques/), ping sweeps, OS detection, and version detection.



**Using Zenmap GUI in windows**

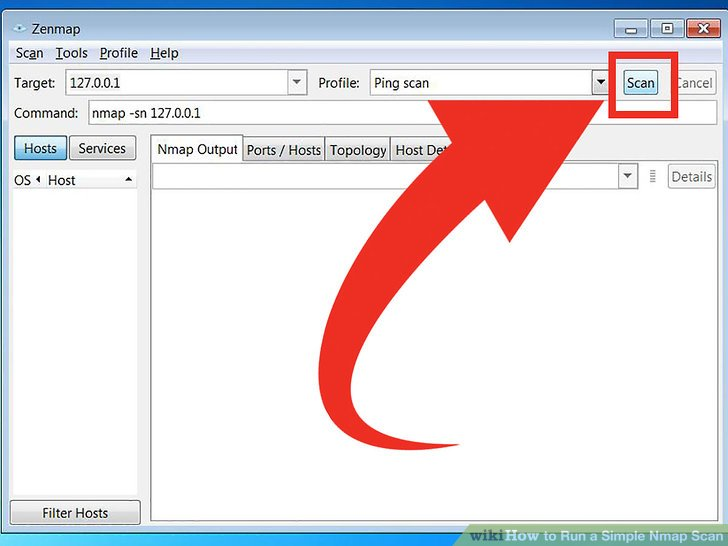
****

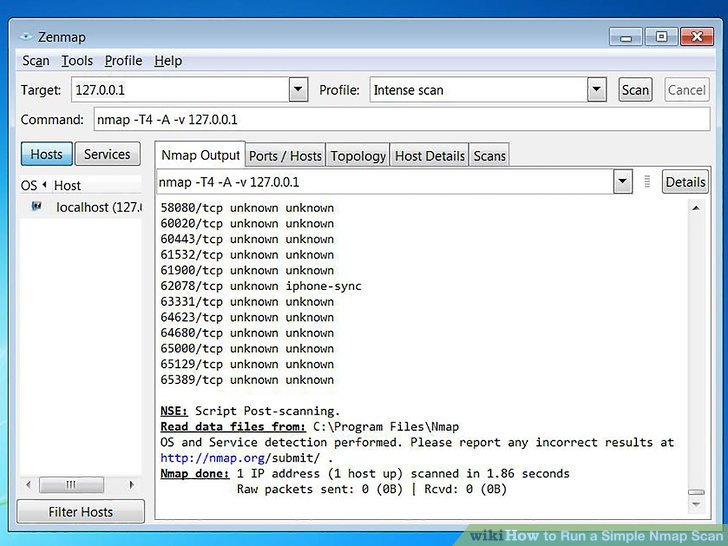
**Enter in the target for your scan. The** Zenmap program makes scanning a fairly simple process. The first step to running a scan is choosing your target. You can enter a domain (example.com), an IP address (127.0.0.1), a network (192.168.1.0/24)

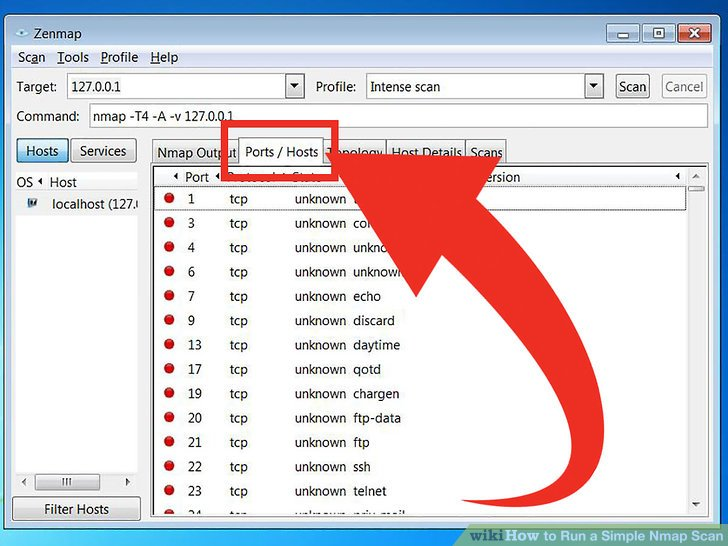


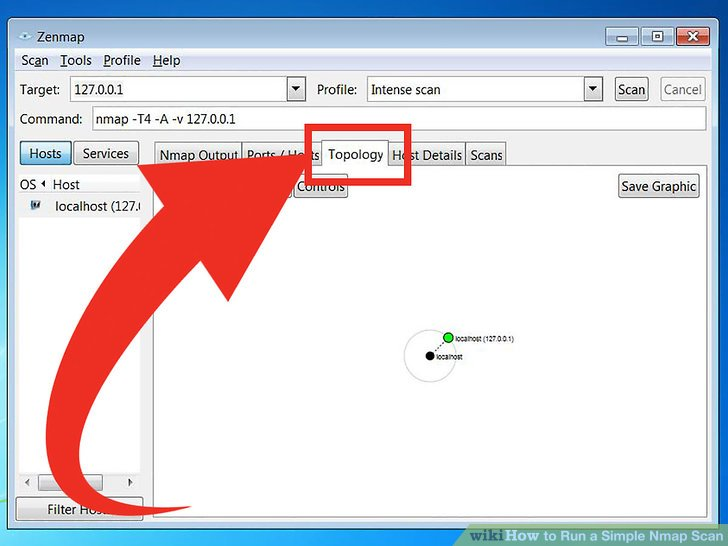
* **Intense scan** - A comprehensive scan. Contains Operating System (OS) detection, version detection, script scanning, traceroute, and has aggressive scan timing. This is considered an intrusive scan.
* **Ping scan** - This scan simply detects if the targets are online, it does not scan any ports.
* **Quick scan** - This is quicker than a regular scan due to aggressive timing and only scanning select ports.
* **Regular scan** - This is the standard Nmap scan without any modifiers. It will return ping and return open ports on the target.

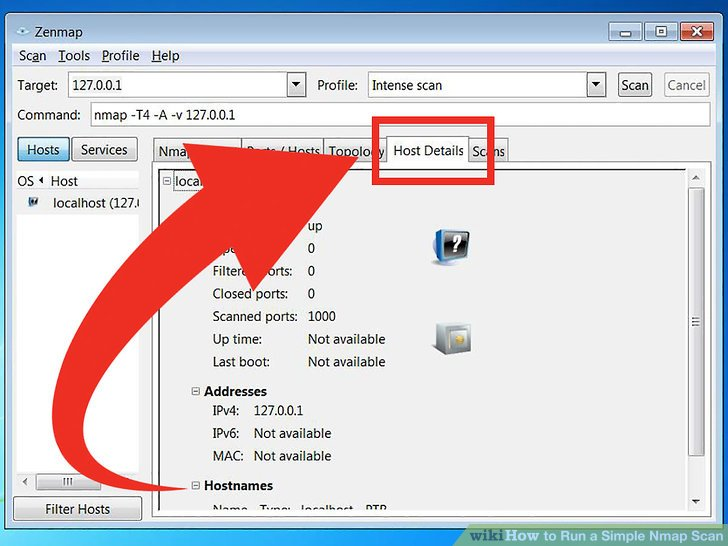
**Click Scan to start scanning.** The active results of the scan will be displayed in the Nmap Output tab. The time the scan takes will depend on the scan profile you chose, the physical distance to the target, and the target’s network configuration.



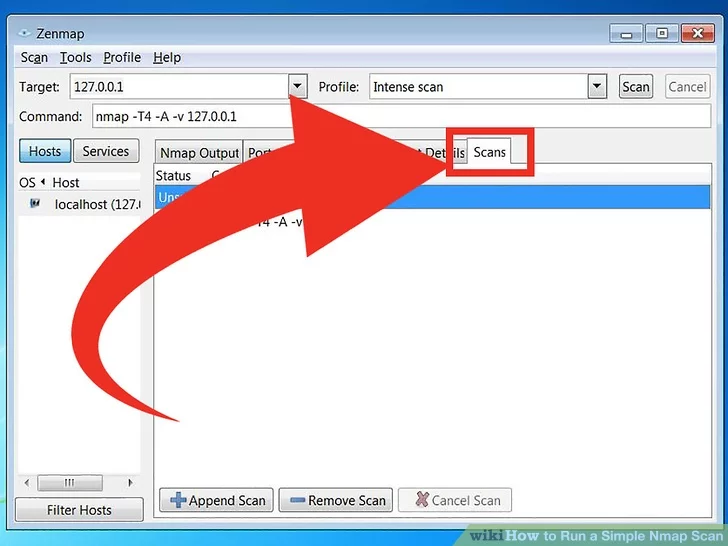








**Scans** - This tab stores the commands of your previously-run scans. This allows you to quickly re-scan with a specific set of parameters.



**Nmap Commands**

**Scanning Techniques**

| **Flag** | **Use** | **Example** |
| --- | --- | --- |
| **-sS** | **TCP syn port scan** | **nmap -sS 192.168.1.1** |
| **-sT** | **TCP connect port scan** | **nmap -sT 192.168.1.1** |
| **–sU** | **UDP port scan** | **nmap –sU 192.168.1.1** |
| **–sA** | **TCP ack port scan** | **nmap –sA 192.168.1.1** |

***Host Discovery***

| **Flag** | **Use** | **Example** |
| --- | --- | --- |
| **-Pn** | **only port scan** | **nmap -Pn 192.168.1.1** |
| **-sn** | **only host discover** | **nmap -sn 192.168.1.1** |
| **-PR** | **arp discovery on a local network** | **nmap -PR 192.168.1.1** |
| **-n** | **disable DNS resolution** | **nmap -n 192.168.1.1** |

***Port Specification***

| **Flag** | **Use** | **Example** |
| --- | --- | --- |
| **-p** | **specify a port or port range** | **nmap -p 1-30 192.168.1.1** |
| **-p-** | **scan all ports** | **nmap -p- 192.168.1.1** |
| **-F** | **fast port scan** | **nmap -F 192.168.1.1** |

***service Version and OS Detection***

| **Flag** | **Use** | **Example** |
| --- | --- | --- |
| **-sV** | **detect the version of services running** | **nmap -sV 192.168.1.1** |
| **-A** | **aggressive scan** | **nmap -A 192.168.1.1** |
| **-O** | **detect operating system of the target** | **nmap -O 192.168.1.1** |

**Scanning Multiple Hosts**

Nmap has the capability of scanning multiple hosts simultaneously. This feature comes in real handy when you are managing vast network infrastructure.

* Write all the IP addresses in a single row to scan all of the hosts at the same time
* nmap 192.164.1.1 192.164.0.2 192.164.0.2

Web links

Installation link for windows

<https://nmap.org/download.html>

Process that happen in windows

<https://www.wikihow.com/Run-a-Simple-Nmap-Scan>

[**https://www.unixmen.com/10-practical-examples-linux-nmap-command/**](https://www.unixmen.com/10-practical-examples-linux-nmap-command/)

**tcpdump** is a data-network [packet analyzer](https://en.wikipedia.org/wiki/Packet_analyzer) computer program that runs under a [command line interface](https://en.wikipedia.org/wiki/Command_line_interface). It allows the user to display [TCP/IP](https://en.wikipedia.org/wiki/TCP/IP) and other packets being transmitted or received over a [network](https://en.wikipedia.org/wiki/Computer_network) to which the computer is attached.

**Tcpdump uses libpcap library to capture the network packets & is available on almost all Linux/Unix flavors.**

**tcpdump** is a packet sniffing and packet analyzing tool for a System Administrator to troubleshoot connectivity issues in Linux. It is used to capture, filter, and analyze network traffic such as TCP/IP packets going through your system. It is many times used as a security tool as well. It saves the captured information in a pcap file, these pcap files can then be opened through [Wireshark](https://www.geeksforgeeks.org/introduction-to-wireshark/) or through the command tool itself.

### **Installing tcpdump tool in Linux**

Many Operating Systems have tcpdump command pre-installed but to install it, use the following commands.

**For RedHat based linux OS**

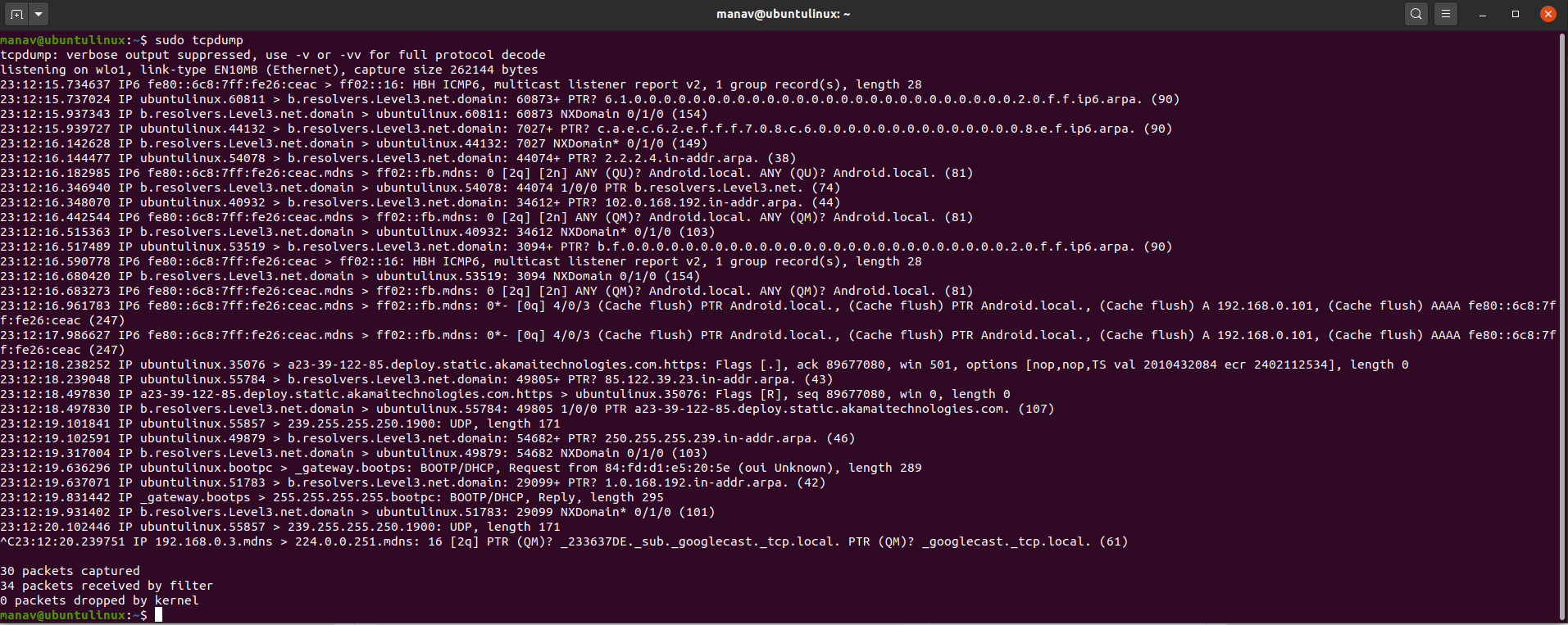
yum install tcpdump

**For Ubuntu/Debian OS**

sudo apt install tcpdump

**1.** To capture the packets of current network interface

sudo tcpdump

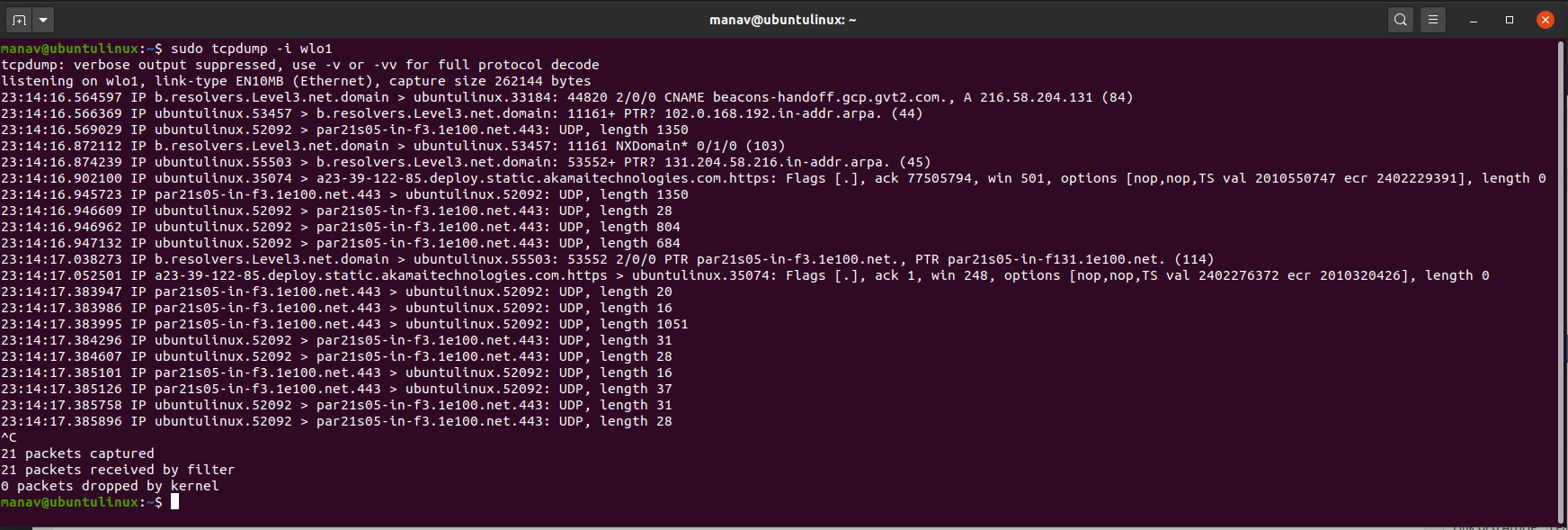


This will capture the packets from the current interface of the network through which the system is connected to the internet.

**2.** To capture packets from a specific network interface

sudo tcpdump -i wlo1

sudo tcpdump -i eno1

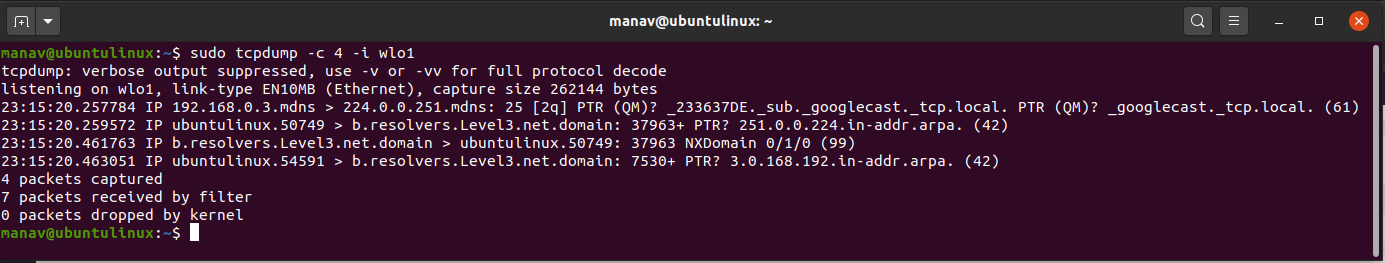


This command will now capture the packets from wlo1 network interface.

**3.** To capture specific number of packets

sudo tcpdump -c 4 -i eno1

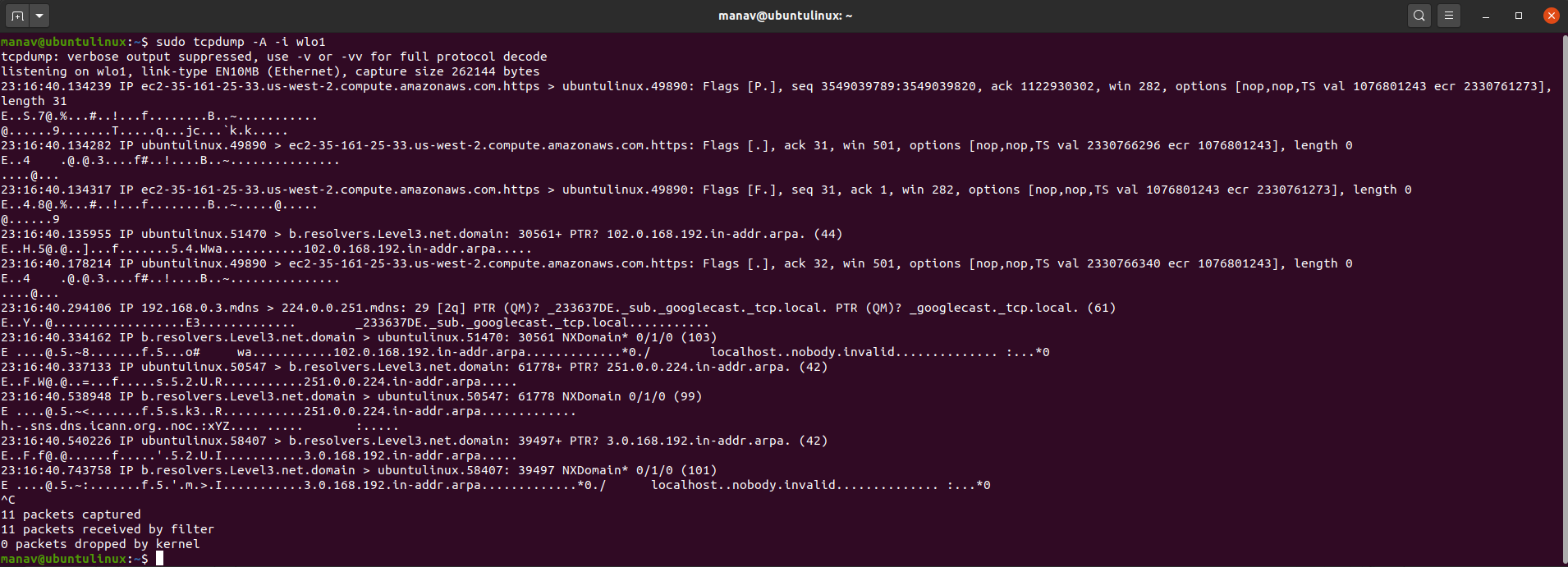
sudo tcpdump -c 10 -i enp2s0



This command will capture only 4 packets from the wlo1 interface.

To print captured packages in ASCII format

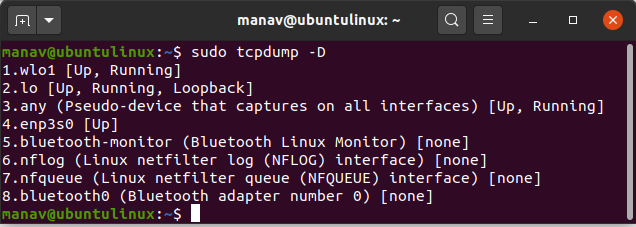
sudo tcpdump -A -i eno1



This command will now print the captured packets from wlo1 to ASCII value.

**5.** To display all available interfaces

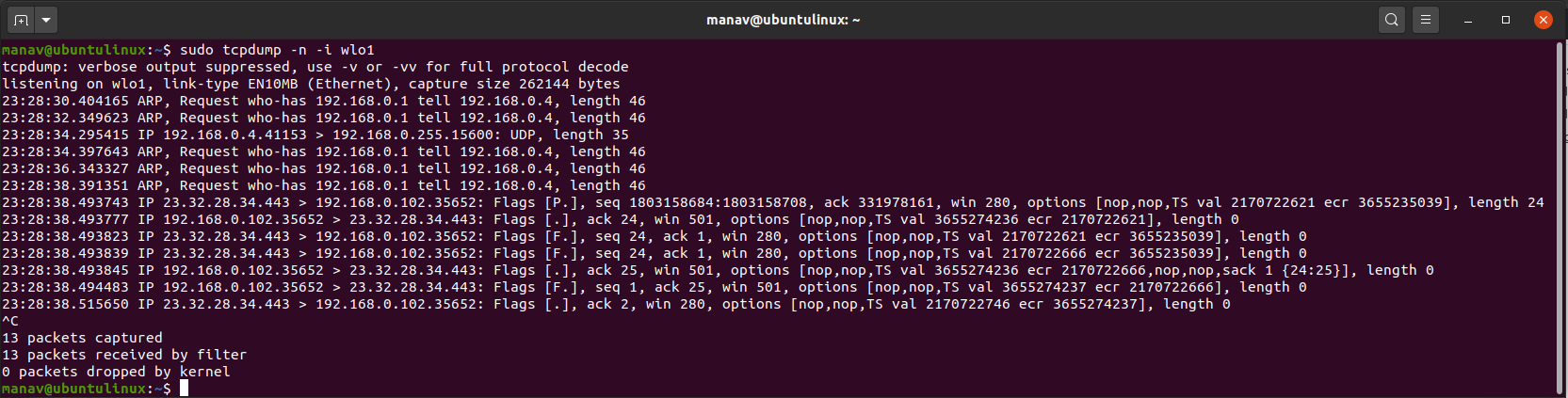
sudo tcpdump -D



this command will display all the interfaces that are available in the system

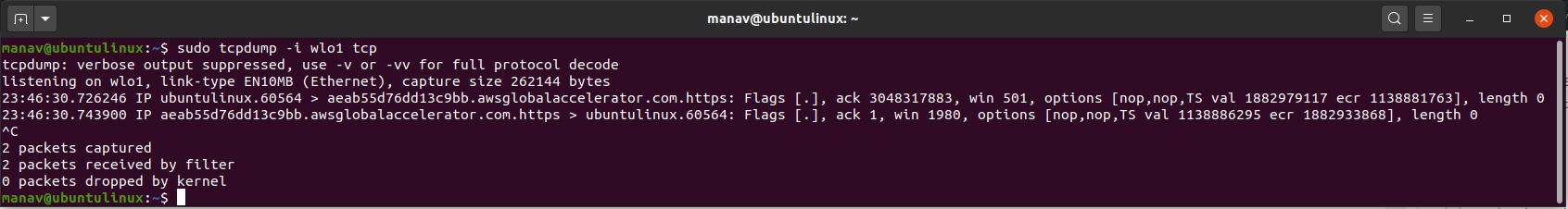
To capture packets with ip address

sudo tcpdump -n -i eno1



To capture only TCP packets

sudo tcpdump -i eno1 tcp



Get all the packets based on the IP address, whether source or destination or both, using the following command,

$ tcpdump host 192.168.1.100

To get packets based on source or destination of an IP address, use

$ tcpdump src 192.168.1.100

$ tcpdump dst 192.168.1.100

Dumpcap is a network traffic dump tool. It captures packet data from a live network and writes the packets to a file. Dumpcap’s native capture file format is pcapng, which is also the format used by Wireshark.

By default, Dumpcap uses the pcap library to capture traffic from the first available network interface and writes the received raw packet data, along with the packets’ time stamps into a pcapng file.

https://www.wireshark.org/docs/man-pages/dumpcap.html

What is Wireshark?

Wireshark is an open-source packet analyzer, which is used for **education, analysis, software development, communication protocol development, and network troubleshooting**.

It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a **sniffer, network protocol analyzer, and network analyzer**. It is also used by network security engineers to examine security problems.

Wireshark is a free to use application which is used to apprehend the data back and forth. It is often called as a free packet sniffer computer application. It puts the network card into an unselective mode, i.e., to accept all the packets which it receives.

Uses of wireshark

Wireshark can be used in the following ways:

1. It is used by network security engineers to examine security problems.
2. It allows the users to watch all the traffic being passed over the network.
3. It is used by network engineers to troubleshoot network issues.
4. It also helps to troubleshoot latency issues and malicious activities on your network.
5. It can also analyze dropped packets.
6. It helps us to know how all the devices like laptop, mobile phones, desktop, switch, routers, etc., communicate in a local network or the rest of the world.

What is a packet?

A packet is a unit of data which is transmitted over a network between the origin and the destination. Network packets are small, i.e., maximum **1.5 Kilobytes for Ethernet packets and 64 Kilobytes for IP packets**. The data packets in the Wireshark can be viewed online and can be analyzed offline.

Functionality of Wireshark:

Wireshark is similar to tcpdump in networking. **Tcpdump** is a common packet analyzer which allows the user to display other packets and TCP/IP packets, being transmitted and received over a network attached to the computer. It has a graphic end and some sorting and filtering functions. Wireshark users can see all the traffic passing through the network.

Wireshark can also monitor the unicast traffic which is not sent to the network's MAC address interface. But, the switch does not pass all the traffic to the port. Hence, the promiscuous mode is not sufficient to see all the traffic. The various network taps or **port mirroring** is used to extend capture at any point.

Port mirroring is a method to monitor network traffic. When it is enabled, the switch sends the copies of all the network packets present at one port to another port

What is color coding in Wireshark?

The packets in the Wireshark are highlighted with **blue**, **black**, and **green color**. These colors help users to identify the types of traffic. It is also called as **packet colorization**.

Features of Wireshark

* It is multi-platform software, i.e., it can run on Linux, Windows, OS X, FreeBSD, NetBSD, etc.
* It is a standard three-pane packet browser.
* It performs deep inspection of the hundreds of protocols.
* It often involves live analysis, i.e., from the different types of the network like the Ethernet, loopback, etc., we can read live data.
* It has sort and filter options which makes ease to the user to view the data.
* It is also useful in VoIP analysis.
* It can also capture raw USB traffic.
* Various settings, like timers and filters, can be used to filter the output.
* It can only capture packet on the PCAP (an application programming interface used to capture the network) supported networks.
* Wireshark supports a variety of well-documented capture file formats such as the PcapNg and Libpcap. These formats are used for storing the captured data.
* It is the no.1 piece of software for its purpose. It has countless applications ranging from the **tracing down, unauthorized traffic, firewall settings, etc**.

For More Information refer to the below web link

https://www.javatpoint.com/wireshark