# Assignment No.: BD5

**TITLE:**
Write a program in Python/ Java/ C++ /Scala using Eclipse to Start/Stop the IDS, View current traffic, view blocked list (IP, Domains), view current firewall rules and unblock users. Create Necessary GUI.

**OBJECTIVES:**

1. To develop problem solving abilities using Mathematical Modeling.

**THEORY:**

An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system.

There are several ways to categorize IDS:

- **Misuse detection** vs. **anomaly detection**: in misuse detection, the IDS analyzes the information it gathers and compares it to large databases of attack signatures. Essentially, the IDS looks for a specific attack that has already been documented. Like a virus detection system, misuse detection software is only as good as the database of attack signatures that it uses to compare packets against. In anomaly detection, the system administrator defines the baseline, or normal, state of the networks traffic load, breakdown, protocol, and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and look for anomalies.
- **Network-based** vs. **host-based systems**: in a network-based system, or NIDS, the individual packets flowing through a network are analyzed. The NIDS can detect malicious packets that are designed to be overlooked by a firewall simplistic filtering rules. In a host-based system, the IDS examines at the activity on each individual computer or host.
- **passive system** vs. **reactive system**: in a passive system, the IDS detects a potential security breach, logs the information and signals an alert. In a reactive system, the IDS responds to the suspicious activity by logging off a user or by reprogramming the firewall to block network traffic from the suspected malicious source.

Difference between IPS and IDS:

| Sr no. | Parameters | Intrusion Prevention System | Intrusion Detection System |
|---|---|---|---|
| 1. | Placement in network infrastructure | Part of the direct line of communication(inline) | Outside direct line of communication(Out of band) |
| 2. | System Type | Active and/or passive | Passive |
| 3. | Detection Mechanisms | 1.Stastical anomaly based detection | Signature detection |

| | | | |
|---|---|---|---|
| | | 2. Signature detection | |
| 4. | Usefulness | Ideal for blocking web destruction | Ideal for identifying blocking attacks |
| 5. | Traffic Requirement | "Original" traffic is required | Traffic replication is required |

Types of IDS:

1. Active and passive IDS:

   An **active Intrusion Detection Systems (IDS)** is also known as Intrusion Detection and Prevention System (IDPS). Intrusion Detection and Prevention System (IDPS) is configured to automatically block suspected attacks without any intervention required by an operator.

   A **passive IDS** is a system that's configured to only monitor and analyze network traffic activity and alert an operator to potential vulnerabilities and attacks. A passive IDS is not capable of performing any protective or corrective functions on its own.

2. Network Intrusion detection systems (NIDS) and Host Intrusion detection systems (HIDS)

   **Network Intrusion Detection Systems (NIDS)** usually consists of a network appliance (or sensor) with a Network Interface Card (NIC) operating in promiscuous mode and a separate management interface. The IDS is placed along a network segment or boundary and monitors all traffic on that segment.

   A **Host Intrusion Detection Systems (HIDS)** and software applications (agents) installed on workstations which are to be monitored. The agents monitor the operating system and write data to log files and/or trigger alarms.

3. Knowledge-based (Signature-based) IDS and behavior-based (Anomaly-based) IDS

   A **knowledge-based (Signature-based) Intrusion Detection Systems (IDS)** references a database of previous attack signatures and known system vulnerabilities. The meaning of word signature, when we talk about Intrusion Detection Systems (IDS) is recorded evidence of an intrusion or attack. Each intrusion leaves a footprint behind (e.g., nature of data packets, failed attempt to run an application, failed logins, file and folder access etc.). These footprints are called signatures.

   A Behavior-based (Anomaly-based) Intrusion Detection Systems (IDS) references a baseline or learned pattern of normal system activity to identify active intrusion attempts. Deviations from this baseline or pattern cause an alarm to be triggered.

Advantages of IDS:

1. It can detect the unauthorized user
2. It can detect password cracking and denial of services
3. It can catch illegal data manipulations
4. Monitors the operations of firewalls
5. Allows administrator to tune, organize operating system audit trails other logs.

Disadvantages of IDS:

1. Host-based IDS works well for a single machine, extremely labor-intensive of monitor multiple machines.
2. If host is compromised, then no more alerts will be generated

3. There is no detection with unknown signatures

Example:

**Examples of Network IDS:**

- SNORT

**Examples of HIDS:**

- OSSEC - Open Source Host-based Intrusion Detection System
- Tripwire
- AIDE - Advanced Intrusion Detection Environment
- Prelude Hybrid IDS

**MATHEMATICAL MODEL:**

Let P be the solution perspective.

P ={ S , E , I , O, F }

S = { Initial state of the IDS, Initialize IP Tables with forward chain replica }

I = Input of the system → { I1 , I2 }
where,
I1 = { IpTable_INPUT }
I2 = { IpTable_FORWARD }

O = Output of the system → { O1 , O2, O3, O4 }
where,
O1 = { list_blocked_ip }
O2 = { display_firewall_rules}
O3 = {Allow_unblock_ip}
O4 = {Allow_IDS_ON/OFF}

F = Functions used → { f1 , f2 , f3 }
where
f1 = { IDS on / off }
f2 = { Blocked_ip_iptables }
f3 = { Unblock_ip_iptables }
f4 = {Display_rules}
E = End state of the system shows successfully handling and updating the IDS and firewall rules of the system.

## IMPLEMENTATION DETAILS / DESIGN LOGIC:

*(Algorithm/Flow Charts/Pseudo Code/DFD/UML diagrams)*

**Algorithm:-**

1. Initialize the IP Tables with forward chain and Input chain.
2. Install open source IDS on the system(In this case PSAD IDS) as follows:
   **sudo apt get install psad**
3. Create a logging of the INPUT and FORWARD chain as follows:
   **sudo iptables –A INPUT –j LOG**
   **sudo iptables –A FORWARD –j LOG**
4. Flush the rules if any made earlier as follows:
   **sudo iptables –F**
5. Display the rules:
   **sudo iptables –L**
6. Explicitly allow all traffic related to an existing connection:
   **sudo iptables –A INPUT –m conntrack –ctstate ESTABLISHED,RELATED –j ACCEPT**
7. Make these rules persistent so that it can't be default after restart:
   **sudo apt-get install iptables-persistent**
   **sudo service iptables-persistent start**
8. Open and change the config file if required:
   **sudo gedit /etc/psad/psad.conf**
   **search for e-mail and domain**
9. Restart the service of PSAD:
   **sudo service psad restart**

**Input :-**

Iptable INPUT and FORWARD chain

**Expected Output :-**

1. Allow IDS on/off
2. Display Blocked IP list
3. Display Firewall rules
4. Unblock IPs

Execute the program with the following commands.

**Note: Make sure you should be sudo user.**

>sudo su
>javac IDS1.java
>java IDS

## TEST CASES:

| Test ID | Description | Input | Expected output | Actual output |
|---|---|---|---|---|
| 1. | List all blocked IPs | INPUT CHAIN | Should list all blocked IPs | Listing all blocked IPs |
| 2. | Start IDS | IDS install | Should start the IDS | IDS got started |
| 3. | Start IDS | IDS not install | Should provide the exception and not able to start IDS | IDS not started |
| 4. | List Firewall rules | FORWARD chain | Should display Firewall rules | Displaying firewall rules |
| 5. | Unblocking the IPs | IP address | Should unblock IP(s) from the IP Table | Unblocked and removed from IP tables |

**CONCLUSION:**

Hence based on the rule of firewall and IDS, we have learnt about the blocking and unblocking the IPs from the IP tables and configuring the IDS.

```
import java.awt.Container;
import java.awt.TextArea;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;


public class IDS1 {

public static String line;
public static JFrame fr;
public static Container c;
public static JTextArea tx;
public static JScrollPane js;
public static JButton ids_on,ids_off,disp_blocked_ip,disp_rules,unblock_ip;
public static JTextField ip;



public IDS1(){
        fr=new JFrame();
        c=fr.getContentPane();
        c.setLayout(null);

        fr.setTitle("Intrusion Detection System Config");

        fr.setBounds(0, 0, 920, 550);

        //components on the frame
        tx=new JTextArea();
        js=new
JScrollPane(tx,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        ids_on=new JButton("IDS ON");
        ids_off=new JButton("IDS OFF");
        disp_blocked_ip=new JButton("Blocked IPs");
        disp_rules=new JButton("Firewall Rules");
        ip=new JTextField("Enter Ip address");
        unblock_ip=new JButton("Unblock Ip");



        //setting bounds
        js.setBounds(5, 20, 900, 400);
        ids_on.setBounds(10, 470, 120, 50);
        ids_off.setBounds(10, 470, 120, 50);
        disp_blocked_ip.setBounds(140, 470, 120, 50);
        disp_rules.setBounds(270, 470, 120, 50);
        unblock_ip.setBounds(410, 470, 120, 50);
        ip.setBounds(410, 520, 220, 50);

        ip.setVisible(false);
```

```
//adding components on the frame container
c.add(js);
c.add(ids_on);
c.add(ids_off);
c.add(disp_blocked_ip);
c.add(unblock_ip);
c.add(ip);
c.add(disp_rules);


//all button's action listeners

ids_on.addActionListener(new ActionListener() {
	@Override
	public void actionPerformed(ActionEvent arg0) {
		ids_on.setVisible(false);
		ids_off.setVisible(true);
		exec_commands("sudo service psad start");
		exec_commands("sudo service psad status");
	}
});

ids_off.addActionListener(new ActionListener() {
	@Override
	public void actionPerformed(ActionEvent arg0) {
		ids_on.setVisible(true);
		ids_off.setVisible(false);

		exec_commands("sudo service psad stop");
		exec_commands("sudo service psad status");
	}
});

disp_blocked_ip.addActionListener(new ActionListener() {
	@Override
	public void actionPerformed(ActionEvent arg0) {
		exec_commands("sudo iptables -L INPUT -v -n --line-numbers");
	}
});

disp_rules.addActionListener(new ActionListener() {
	@Override
	public void actionPerformed(ActionEvent arg0) {
	//exec_commands("sudo iptables -N TRAFFIC_ACCT");//own traffic chain in order to
avoid changes in firewall rules
	//exec_commands("sudo iptables -I FORWARD -j TRAFFIC_ACCT");//forwarding all
traffic to my created chain
	//exec_commands("iptables -A TRAFFIC_ACCT -p tcp && iptables -A TRAFFIC_ACCT -p
ip && iptables -A TRAFFIC_ACCT -p icmp");
		exec_commands("sudo iptables -L");

	}
});

unblock_ip.addActionListener(new ActionListener() {
	@Override
	public void actionPerformed(ActionEvent arg0) {
		String response = JOptionPane.showInputDialog(null,"Enter IP address",
```

```
                                        JOptionPane.QUESTION_MESSAGE);
                        exec_commands("sudo iptables -D INPUT -s "+response+" -j DROP");

                }
        });

        fr.setVisible(true);
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

//Main method
public static void main(String arr[]) throws Exception{
IDS1 ids=new IDS1();
}

//method for execute the commands
public void exec_commands(String cmd){
try {
        Runtime rt = Runtime.getRuntime();
        //Process pr = rt.exec("cmd /c dir");
        Process pr = rt.exec(cmd);

        BufferedReader input = new BufferedReader(new InputStreamReader(pr.getInputStream()));

        String line=null;
        tx.setText("");
        while((line=input.readLine()) != null) {
            System.out.println(line);

            //display cmd output on textarea tx

            tx.append(line+"\n");
        }
        //int exitVal = pr.waitFor();
       // System.out.println("Exited with error code "+exitVal);

   } catch(Exception e) {
        System.out.println(e.toString());
        e.printStackTrace();
   }
}

}
```