Implementation of a Prototype for a Real-Time Monitoring, Detecting and Alarming System in Computer Networks

GRADUATE PROJECT REPORT

Submitted to the Faculty of
The School of Engineering & Computing Sciences
Texas A&M University-Corpus Christi

Corpus Christi, TX

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

By

Keerthi Reddy Muthyala
Spring 2015

Committee Members

Dr. Mario Garcia
Committee Chairperson

Dr. David Thomas
Committee Member
ABSTRACT

Network Security provides security to the network and thereby protecting it from different types of attacks occurring in the network, and it also monitors the operations performed on the network. It comprises certain policies that are handled by the main authorities of the network, to counteract and these authorities examine the illegal access, modification or denial of a computer network and its resources. Similarly, security checking is conducted on the network to assess the status of the network. This is a process of assessing the security of the network components in order to discover the vulnerabilities, before the attackers can exploit them.

The main objective of the proposed system is finding the vulnerabilities and alarming the responsible entities, which is achieved with an open source software, “Snort” and a web application called Alarm System. Snort is a Network Intrusion Detection System that detects the threats or vulnerabilities existing in the system and also alerts the administrators. The results obtained from Snort are retrieved from the local host using Java language and are collected into MySQL database. The results are then executed by the Apache Tomcat server which is present in the developed alarm notify web application. The entire concept of the project is combining the Intrusion Detection System with the alarm system, so that the attacks detected by Snort are used by the administrator for sending an alert message to the responsible entities in the form of an email attachment.
TABLE OF CONTENTS

Abstract ........................................................................................................................................ ii

Table of Contents ..................................................................................................................... iii

List of Figures ........................................................................................................................... v

1. Background and Rationale ...................................................................................................... 1
   1.1 Introduction ...................................................................................................................... 1
   1.2 Security ............................................................................................................................ 3
      1.2.1 Firewalls ................................................................................................................ 3
      1.2.2 Vulnerabilities ........................................................................................................ 3
   1.3 Intrusion Detection System .............................................................................................. 5
      1.3.1 Functions of IDS .................................................................................................... 6
   1.4 Intrusion Detection Technologies ..................................................................................... 7
      1.4.1 Host-based Intrusion Detection System (HIDS) ..................................................... 7
      1.4.2 Network Intrusion Detection System (NIDS) ......................................................... 8
   1.5 Snort ................................................................................................................................ 9
      1.5.1 Packet Sniffer .......................................................................................................... 9
      1.5.2 Signatures ................................................................................................................ 10
      1.5.3 Alerts ...................................................................................................................... 10
      1.5.4 Logs ....................................................................................................................... 11
   1.6 Snort as a NIDS ............................................................................................................. 11

2. Previous Research and Scope ............................................................................................... 13
   2.1 Previous Research .......................................................................................................... 13
LIST OF FIGURES

Figure 1.1 Intrusion Detection System ................................................................. 6
Figure 1.2 Network Intrusion Detection System .................................................... 8
Figure 1.3 Example of an Existing Signature for Cross Site Scripting ..................... 10
Figure 2.1 Vulnerability Statistics Reported over the Years ................................. 14
Figure 2.2 Percentage of Incidents Reported in 2013 ........................................ 16
Figure 3.1 The Proposed System Architecture ..................................................... 18
Figure 3.2 Flow Chart Diagram of the Proposed System .................................... 21
Figure 3.3 Architecture of Snort ........................................................................... 22
Figure 3.4 The Sample of a Rule Header and Rule Options .................................. 25
Figure 3.5 Rules configured in Content-replace.rules ....................................... 25
Figure 3.6 Rules configured in local.rules ............................................................ 26
Figure 3.7 Rules configured in myrules.rules ....................................................... 27
Figure 3.8 Screen shot of Rules to Detect DDoS flood attacks ............................ 27
Figure 3.9 Use Case Diagram .............................................................................. 32
Figure 3.10 Class Diagram ................................................................................ 33
Figure 3.11 Sequence Diagram .......................................................................... 34
Figure 4.1 Login Page of the Alarm System ......................................................... 36
Figure 4.2 Entering the user name and password ............................................... 37
Figure 4.3 Command prompt Displaying the Environment for Snort .................... 37
Figure 4.4 Initializing and Configuring Snort ....................................................... 38
Figure 4.5 Configuring the Snort rules in local drive .......................................... 38
Figure 4.6 Console Showing the Execution of Rules ........................................... 39
Figure 4.7 Interface of MySQL Database............................................................... 40
Figure 4.8 Execution of Alerts in Apache Tomcat Server....................................... 41
Figure 4.9 Displays the alerts on home.jsp of web application................................. 42
Figure 4.10 Shows the Excel file for Displaying the alerts ........................................ 42
Figure 4.11 Interface for Sending the email attachment............................................ 43
Figure 4.12 Email is Sent Successfully................................................................. 44
Figure 4.13 Report containing Alerts in the Gmail inbox ....................................... 44
Figure 4.14 Alert updated successfully as Resolved ............................................... 45
Figure 4.15 After Resolving the alert, exit Snort................................................... 45
Figure 5.1 Shows the Execution of SYN Flood attack ............................................. 47
Figure 5.2 Shows the Alerts for SYN flood on home.jsp ....................................... 48
Figure 5.3 Screenshot for MySQL interface............................................................ 49
Figure 5.4 Screenshot for MySQL root login attempt ............................................. 49
Figure 5.5 Screenshot of Internet Explorer opening multiple tabs ............................. 50
Figure 5.6 Screenshot of the Detected Virus .......................................................... 51
Figure 5.7 Screen Shot for using Unauthorized Websites ....................................... 51
1. BACKGROUND AND RATIONALE

1.1 Introduction

Security is a major concern in the present era of technology and science. All the reputed organizations aim for achieving the security because they will be dealing with many confidential affairs that are linked with many countries. So, they are ready to spend a lot of money for gaining the security of the company. The intruders take this as a useful opportunity and they will try to defame the reputed companies. In order to avert their attacks, many tools and software are available in the market for example, Firewalls, vulnerability tools and many Intrusion Detection Systems. The main reason behind all these attacks is, the systems that are present in the network of the organizations are connected to the internet. As, the internet is used by many people in the world, it is not very secure for the private organizations that are interlinked with the internet. The problems that are caused, arise as vulnerabilities, leaving a harmful effect on the organization [2]. WWW and email are considered as the widely used application programs that are associated with the network, these applications will be running on all the end systems. The predefined protocols direct the data that is exchanged between the computers. These protocols like TCP and IP controls both sending and receiving of information within the internet [8]. Many attacks like Spoofing, Sniffing, and Denial of service create a huge loss to the company.

The main objective of this project is to implement a prototype for monitoring, detecting and alarming system to enhance security in computer networks. This can be achieved by using a software called “Snort” which helps in monitoring, diagnosing and
detecting intrusions. A web application for alarm system is also developed in addition to
the NIDS. Snort is free to use as it an open source application. The recent advancements
made in the software assisted in discovering the attacks and this helped the administrator
in safeguarding the network. Research has been conducted on Arbor Networks, the
statistics of the DDoS attacks account for 20Gbps in the beginning of 2014 and they are
more when compared to that of the previous year [9]. In 2014, an attack occurred on the
most popular shopping web-site i.e., E-Bay, it announced that the hackers acquired the
personal records of many users and that count was estimated to be 233 million [9]. That
is the main reason for the growing demand of network security.

The importance of the project:

- The proposed system helps in assessing the security of the network using an open
  source software called “Snort”.

- Snort is a packet sniffer and also a Network Intrusion Detection system that
  observes the traffic on the network and examines all the packets passing through
  the network.

- A virtual network is created to create attacks in the network and then detect those
  attacks using Snort.

- The attacks that are generated are SYN Flood attack, Database root login attack,
  Virus detection and using unauthorized sites. The Snort then captures the alerts
  and displays it in the web application.

- The prototype designed here, is integrating Snort along with the web application
  which is developed using JAVA, HTML, Servlets and JSP .It provides an easy
  way of access to the administrators and the users.
• The report generated by Snort is retrieved from the local host, collected in MySQL database and then displayed on the interface of web application with the help of Tomcat server. The report is then sent as an email attachment to the users.

1.2 Security

Generally, security means giving protection either to a system or a network or to an application. It must protect the company from the outsiders who are trying to gain access to the network. If the security system collects information about the attacks occurring in the network, it can try to stop those attacks and that increases the security of the network. Generally, security can be developed by many tools and techniques available in the market. Those softwares and tools are open source and some of them are licensed versions which are to be bought and installed. Firewalls, Vulnerability tools and Intrusion Detection Systems come under the means of providing the security to a network.

1.2.1 Firewalls

They come under the basic components of the network that implements the security. Firewalls behave like a filter that filters all the packets which are going in and out of the network [16]. The main aim of the firewall is to verify each and every single packet that is going through it. It is basically placed at a point in the network, which connects the outside world with the inside network and if it finds any packet to be vicious then it drops that particular packet.

1.2.2 Vulnerabilities

Vulnerability is considered as a weakness that is existing in the system hardware or software, and helping the attacker to enter easily into the system which causes a
potential risk to the system. An increase in the number of vulnerabilities, viruses and internet worms leads to the development of a technology that results in an assessment of the vulnerabilities. Checking the vulnerabilities manually is the easiest way to analyze the vulnerability of the network because a person can log in into the system and can check all the applications that are installed on the systems along with their versions and services that are running on them and can also see the security status of the system. But this kind of approach works only with single hosts. So, for large and complex networks, automated vulnerability assessment tools [1] [3] are used because they are more effective and efficient. The whole process is an automated process so, the vulnerabilities can be easily scanned [5]. All the vulnerability scanners work on the principle of automatically connecting to the intended system with the goal of finding the vulnerabilities in the system. Some examples of the tools that assess the vulnerabilities are Nessus, OpenVAS, and Core Impact.

For the detection of vulnerabilities, network scanning is done and this is called as network security assessment. So it keeps on scanning the entire network for the detection of vulnerabilities and this can assess the status of the security of a system. An assessment cannot be performed easily. So, the network assessment [6] is understood in four phases. These phases are described below:

Reconnaissance: Gathering fundamental and valuable data about system, for example, domain names, individual contacts, IP addresses and so on.

Network Scanning: Scanning of the network is done to detect the hosts that are alive and also the ports that are open to vulnerabilities.
Vulnerability assessment: Considering all the results obtained from the above scanning phase, the vulnerabilities present in the network are determined [6] [7].

Exploitation: The vulnerabilities found are exploited to know their effect on the network.

1.3 Intrusion Detection System

The Intrusion Detection System manages the security of both the systems and the networks. These have the information about each and every computer in the network and so it can easily identify the violation of any security policies by the attacker. The attacks made by the attackers can be in any way, i.e., from the attackers present in the organization or from the attackers who are present outside the organization. This IDS uses the technique of vulnerability assessment because this lessens the burden for IDS in scanning the whole network. An IDS is generally called as passive monitoring system [13] because it detects the harmful activities being performed but they do not concentrate more on preventing those activities.

Figure 1.1 explains the monitoring of traffic by IDS which is coming from the internet. It contains servers, a switch, a firewall, and an intrusion detection system. It monitors the traffic coming from the internet.
This system acquires the packets and by using those packets, it can react to any dangers or threats in the network and can also detect the malicious activities with the help of various signatures [23]. But a small flaw occurs with the IDS while it is trying to detect the threats, i.e., it may allow some harmful entries into the network without even recognizing that a malicious entity has entered. The IDS examines the replica of the observed traffic other than the actual packet of data. The main disadvantage with this system is that it cannot stop the harmful traffic coming from single attacks because it takes time for the IDS to respond to the activity that has occurred. They can be placed at a point on the network so that they should be in a position to cover the entire network. So, an IDS must be used with Firewalls or any other networking devices like routers [13].

1.3.1 Functions of Intrusion Detection system

The major functions of Intrusion Detection System are:

- This system keeps on checking the network and understanding the activities performed by the user and the computers.
• IDS examines the configuration of the computers in the network and also checks for the detection of vulnerabilities.

• Evaluating the trustworthiness of the files.

• This has the capacity to identify the attacks occurring on the network.

• If any changes in the network, then the IDS can easily track the abnormalities.

• Keeps on monitoring to prevent the violation of policies.

The main aim behind the development of these systems is, the detection of the expanding number of attacks on computer networks. Recently these attacks occurred on the Pentagon, United States Defense Department and the White House. Recent statistics state that, hackers are attacking many social networking sites like the Yahoo! Mail. It states that the services provided by Yahoo like emails are being hacked in the month of January and the number has reached 273 million users. On the other hand, the most disgusting part is that the emails have not yet been released [12].

Recently in 2014, an attack has been reported inside the organization, the company that suffered was AT&T and some personnel hacked the social security numbers of the users and this hack lasted for two weeks [12]. This is the main reason for the intrusion detection systems to rule the today’s world.

1.4 Intrusion Detection Technologies

1.4.1 Host-based Intrusion detection System (HIDS)

A HIDS is same as the IDS, it is a software that is being set up on the hosts, and it scrutinizes the traffic that goes in and out of a particular computer in which the software is installed [13]. For example, any changes to the word document or some key files in the
computer can be identified by HIDS, it keeps on examining the whole system for any overwriting of the files. HIDS is more adaptable than NIDS.

1.4.2 Network Intrusion Detection System (NIDS)

In the proposed prototype, Snort tool is used which is an open source Network Intrusion Detection System. So, Network Intrusion Detection System generally monitors the network traffic in order to find the entry of any intruder into the network. Apart from monitoring the traffic, which is going in and out of the network, the NIDS can also scan the server log files [13]. If, the system finds any packet to be malicious then it sends an alert message to a server which maintains a log of all these files. Then later it analyzes the results to check if they lead to any kind of threat.

Figure 1.2 describes the network intrusion detection system. All the systems and sensors are connected to the untrusted network. So, any intrusions occurring through that network are easily identified by NIDS.

![Figure 1.2 Network Intrusion Detection System [16]](image-url)
1.5 Snort

In order to implement this prototype, Snort is being utilized in the development of the project. This came into existence because of the work of Martin Roesch in the year of 1998 [18]. This software is now developed by Source fire which is owned by Roesch. In the year of 2009, Snort was declared as an open source tool. The operation of Snort is from the command prompt and now it has been deployed in other applications and platforms. It is recognized as one of the best IDS tools available in open source which is of free cost to download. When the Snort has been developed, it was designed to work as a packet sniffer, but later it is used in many scenarios that relate to ID systems [18]. Snort contains customized rules that are configured and then later followed to execute a particular functionality. Snort also generates alerts to warn the administrator about the threat. This has been downloaded by many users and its importance is growing day-to-day. This Snort can run on many operating systems like Windows, Linux, and FreeBSD and also on Solaris.

1.5.1 Packet Sniffer:

Packet sniffer keeps on auditing the network by performing various kinds of troubleshooting techniques, to examine the security level of the network. The packet sniffer is a software program which can view all the data passing through the network and the data passing out of the network [18]. The network interface of the packet sniffer is a promiscuous mode, in which each and every single packet is being analyzed by it. Snort acts a packet sniffer and it is called a command-line sniffer because it is executed from the command line and is different from the other tools. The interface it provides is
in a quick readable format, and the summary report it generates is also very useful for assessment.

1.5.2 Signatures

A signature is an arrangement that is viewed inside of a packet, containing data. The location of signatures inside a data packet can be anywhere, based on the location the nature of the attack is identified. Similarly, one signature can be used to identify numerous attacks occurring in the network. For example, a signature for detecting the cross site scripting is given in Figure 1.3.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg: "WEB-MISC cross site scripting attempt"; flow:to_server, established; content:"<SCRIPT>"; nocase; classtype:web-application-attack; sid:1497; rev:6;)
```

**Figure 1.3 Example of an Existing Signature for Cross Site Scripting**

These signatures are used in intrusion detection systems that are based on signatures. Here the packets that are sent are compared to the rules which are predefined. Sometimes, there can be innumerable matching of the patterns found in the network. The NIDS which is used in the proposed system i.e., Snort has many signatures and they can be created numerously on our own depending on the threat or attack [21].

1.5.3 Alerts

If any harmful activities occur in the network and if the administrator is unaware of it, then the intruder can steal all the confidential information from the network. That is the reason, the Intrusion detection systems come with an alarm [21] and this alarm generates an alert which warns the administrator about the malicious activities in the
network, so that he can take an action against it. The main important aim of IDS is to eliminate the false alarms, which are sometimes caused by any changes in the system [21] and the alerts can be of any kind like displaying a pop-up window, blinking on the screen or sending the report. The Snort which is being used in the proposed system can bring about numerous alerts.

1.5.4 Logs

The software which is used in the proposed system has a directory to store all the log files. It is saved in the snort directory. As Snort is executed from the command line, it is possible to change the destination of the logs, in any place. All these log files can then be changed in any format like the normal format or the binary format [21].

1.6 Snort as a NIDS

In the market Snort is basically considered as the best solution for NIDS hence it has been used in many companies. The following facts prove that the Snort is apt as a NIDS [18].

Cost:

In considering the cost of this tool, it is very much less because of its open source feature. Most of them prefer this because everyone will be looking for cutting the cost.

Stability, speed and robustness:

As the network bandwidths are increasing day-by-day, the developers are very careful and so they developed this as a lightweight, robust and a fast application. The statistics of the past years prove that Snort has not been crashed many times. It is very stable because it is being developed to opt to the ruinous environments.
Flexible:

Snort is designed in such a way that it is used in small networks as a sniffer and in larger networks as a very powerful intrusion detector. That is the reason, it is said that it can embrace to any circumstances.
2. Previous Research and Scope

2.1 Previous Research

All the previous research is adopted from different papers cited below. The increase in the advancement of technology has led to drastic changes in the computing environment [7]. This is because previously several measures are taken to prevent the occurrence of attacks on the networks, but those countermeasures couldn’t help in solving the problems. In order to know the cause of the attack, it is more important to know if the attacks have previously occurred on the network or not. For numerous goals including security in explicit, attaining a correct, most recent and clear perspective of an information system is very important. All of these depend mostly on the setup of the software on the end systems. It is hapless that these softwares are not compatible with some hosts. Recently, many tools and software are available in the real world for Intrusion detection systems. With the help of such kind of software, an administrator can easily secure the network, as they provide a detailed report of the intruders trying to attack the network and also if anyone is willing to make any changes to the network are also caught.

J.A.P Marpaung, M. Sain & Hoon-Jae Lee [14] declared that Intrusion detection has been considered as one of the factors in the whole family of anti-intrusion techniques. They originally state that, it is better to prevent an intrusion from occurring rather than go for various kinds of tests to detect whether any virus has been penetrated into the network. So, because of this a dire need for Intrusion Detection approaches has increased.
Y. Yan [17] proposed different intrusion detection approaches. He explains the NIDS as, a system that helps in identifying the traffic that is entering into the network for malicious information. The main reason for intrusion is the existence of vulnerabilities in the network, by going through the statistics it is understood that IDS is most essential in the network.

From figure 2.1 it is clear that the vulnerabilities kept on increasing day by day. There is an increase in the vulnerabilities in 2010, it is 1887 and there is a decrease from 1492 to 1488 from 2011-2012, and in 2014 it is increased to 1705. This is because of the development of some advanced Intrusion Detection systems, and with the advancement in technologies, the vulnerabilities are being caught.

![Figure 2.1 Vulnerability statistics reported over the years [24]](image)

G. Sathya & K. Vasanthraj [20] used different techniques for examining the network. Monitoring of the network is an efficient way of understanding the intrusion
detection systems rather than following the systems that are based on audits and this is considered as the most traditional approach (audit-based). Traditional methods are simple, so they might not withstand the attacks of the recent years. So, more complicated mechanisms of Intrusion Detection are used. It is very difficult to depend completely on the trails of the audits performed.

M. Maatta and T. Raty [15] proposed that advanced methods are being used and they mostly collect the data from the multiple operating systems and the host systems instead of waiting for the data to be acquired from the audit trails of the operating systems. If the attacker knows that some audits are done and he is being monitored this results in preventing the intrusion detection and causes an inability to determine the kind of attack occurred.

L. Lahoti, C. Chandankhede & D. Mukhopadhyay [21] came up with a network monitoring system and this always executes as a passive entity and carefully observes the actions of an intruder. So, if a new entity is entering the network then it should be carefully observed that it does not create any harmful effects on the system. The statistics state that the overall growth of the network performance is changed from 5 to 20 percent when it is related to the audit based systems [20].

From figure 2.2 Intrusion and Intrusion attempt account for 17.88 %. Out of all these, the fraud represents a total of 49.93 % which is the highest of all. The fraud mostly accounts for Phishing, Nigerian Scams and so on. All these statistics are given by MYCERT.
2.2 Existing System

There are many systems for finding vulnerabilities and intrusions, but these systems are executed in a totally different way. K. Novak [4] tested countless vulnerability tools and a vulnerability scan has also been performed [7] to discover the effectiveness and execution. They used to confide on the decision of just one tool. So, K. Novak explained that, one tool is not acceptable to decide on the existence of vulnerabilities and so he choose to use multiple tools. Initially, when they first started using the tools, the tools were used serially one after the other and are analyzed independently. The output of these tools are gathered and later evaluated manually. This process took a lot of time. So an automation in the approach has been recommended to save the time. With the drawbacks of that system our proposed system came into existence.
2.3 Scope:

The main scope of the project is implementing a Prototype for Monitoring, Detecting and Alarming System [11] to enhance security in Computer Networks. So, the proposed system makes use of Snort to perform the required functionalities and operations on the network. The development of web application helps in assessing the security of the network and finding the vulnerabilities of the network. Snort maintains a log of all the files and generates an alert if any threat is found in the network. The alerts are captured by MySQL database and then processed by Apache Tomcat Server. These alerts are displayed in the interface of web application. Then the alarm system is going to use the report in sending it to the responsible entities through email as an email attachment. The proposed implementation should be flexible enough such that it should be able to withstand the intrusions occurred on the Network. Previous studies have shown that one way of detecting the intrusions is not sufficient, hence more sophisticated means of mechanisms detecting the attacks are required.
3. System Design and Architecture

3.1 System Design

In the proposed system, a Real time Monitoring, Detecting and Alarming system is developed. Figure 3.1 shows the architecture of the proposed system. Here a virtual network consisting of three systems is created manually. All the systems in the network are connected by bridging the networks between them. One is the actual physical machine (Windows 8) and two are virtual machines. One virtual machine used is Windows Server 2008 and the other one is Windows 7.
The network is being created to perform the attacks on the physical machine using the two virtual machines and the attacks are being detected by the Snort, present in the physical machine. The attacks that are created in the virtual machine are SYN flood attack, Database Root Login Attempt and Detection of virus. The proposed system uses an existing software called “Snort” for monitoring the network. The alerts generated by Snort may contain the attacks that are occurring, and they are retrieved from the local host and then captured in the internal capture log server. Then MySQL database uses the data from log server and then stores it in the form of tables. MySQL sends this tabular data to Apache tomcat server which is connected to the alarm system (web application). The web application then shows the alerts on home.jsp and then the alerts can be copied and saved in an excel file.

The administrator then sends the attachment of the report, to the users through Email. For the development of a log server and the alarm notify system, the languages used are JAVA HTML and JSP.

3.2 Flow of Execution:

Figure 3.2 shows the flow of execution of a real time monitoring, detecting and alarming system using Snort and a developed web application.

- Open the Alarm Notify System web application. The administrator then logs in into the system.
- Then enable Snort intrusion detection system in the system.
- As the rules are written in myrules.rules file in c:\Snort\rules. Snort uses those rules to capture any attacks or vulnerabilities existing in the system.
• As the snort is executing the rules, the log server is programmed to capture the alerts generated by Snort.

• Then the database stores the alerts coming from log server in the form of tables.

• The Apache Server then collects the data stored in database, to display it in home.jsp, in the form of a table with column names alert no., alert date, alert type, alert message, alert status and resolve alert.

• The administrator then copies the alerts and pastes them in an excel file.

• The alerts generated by Snort, contain the vulnerabilities, malicious content used like pornography, terrorism, malware, virus and also the security attacks if detected.

• This report is emailed to the responsible entities as an email attachment.
Figure 3.2 Flow chart diagram of the Proposed System

1. Start
2. Enable Snort Intrusion in physical machine
3. Configure alerts in rules
4. Opens Alarm System
5. Logs in as administrator
6. Runs commands in Snort on physical machine
7. MySQL captures the alerts and sends it to the alarm system
8. Monitors the alert again
9. Generates attacks using VM
10. Alarm system displays the alerts in tabular format
11. Admin sends the alerts to users through email
12. Alert resolved
   - Yes: Displays “Alert is resolved”
   - No: Alerts monitored again
13. End
3.3 Architecture of Snort:

Snort is a good sniffer which is used in Network Intrusion Detection. The Proposed System makes use of Snort, so the architecture of Snort is clearly explained in figure 3.3. This is a sensor software to sniff the changes in the network and is installed on the server machine which is located in the network to examine the traffic in the network.

![Figure 3.3 Architecture of Snort](image)

From figure 3.3 the functionalities of each section are explained below

- Packet sniffer: This packet sniffer is a software program that can view all the data that is passing through and network and the data that is passed out of the network.
- Pre-processor: The major work of pre-processor is to use the plugins to test the packets, to see if these packets are showing a positive behavior.
• Detection engine: The incoming packets are used by the detection engine and then executes them with some bunch of rules. These rules are comprised of the rule header and rule option [19]. If, any packet is found to be not following the rules then all those packets are neglected. The detection engine in Snort compares each rule with the rule sets and then decides whether to process or not.

• Alerting and logging: If a match is found in the rules of the packets with the signature then it triggers an alert.

3.4 Step by Step process of project development:

The following describes a brief overview of the step by step process of the proposed system.

This project is implemented using 4 modules. They are

1) Configuring Snort
2) Creating customized rules for Snort.
3) Creating the attacks using virtual machine.
4) Developing an Internal capture log server.
5) Developing a web application
6) Testing the configured rules using the created attacks.

1) Configuring Snort:

• Snort is an open source Intrusion Detection System, which configures and captures all the traffic flow inside the network.

• In order to configure the Snort rules, the main goal is to identify the IP addresses of the network for which the in and out flow of the traffic is captured.
• Next the path of the log is also specified, the log is the place where the generated files are stored.

• Snort functions on Network layer, application layer along with transport layer i.e. on TCP/UDP.

• Snort is installed on any operating system environment like Windows, Linux and BSD. WinPcap 4.1.3 is downloaded and installed to make the Snort work on any operating system.

• The configuration file for Snort is located in snort.conf configuration file [19].

2) Creating customized rules for Snort:

• Snort provides the users to write their own customized rules in rules folder present in c:\Snort\rules.

• Rules are generally written by already established signatures.

• The execution of the rules result in generation of alerts. After the rules are created, they are tested on the console by giving the command snort -i3 -s -l c:\snort\log\ -c c:\snort\etc\snort.conf.

• Alerts are configured based on the specifications of the system and the vulnerabilities that may possibly occur in the system.

• The rules are based on the protocol provided like TCP, UDP, ICMP and IP.

Figure 3.4 shows the sample of a rule header and the options of the rules. Here in the rule the first word can be alert the protocol, drop the packets or log the report. Those are the actions that are to be performed by the rule. Then next, the IP address of the home network and the port number then followed by an arrow which specifies the direction. After the arrow the IP address and port number of the external network is given. Next
comes the message that should be displayed on the console, related to the alert. Sid is the Snort identification number to differentiate the rules with one another and rev is the revision integer to update the alerts modified.

![Diagram of rule header and rule options]

**Figure 3.4 The Sample of a Rule Header and Rule Options [19]**

Figure 3.5 shows the rules that are configured in content-replace.rules. This rule searches for the patterns with the words containing words like: “porn”, “child Pornography”. Alerts Internet Protocol if any words like porn or child pornography are found.

```
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# CONTENT-REPLACE RULES
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

alert ip any any -> any any (content:"porn"; msg:"porn word matched"; sid:100005;)
alert ip any any -> any any (content:"child pornography"; msg:"child porn word matched"; sid:100006;)
```

**Figure 3.5 Rules Configured in Content-replace.rules**
Figure 3.6 shows the rules that are configured in local.rules file in Snort folder. Here the rules are one of each kind. Different categories of rules can be written in rules file.

```
# LOCAL RULES

#alert icmp any any -> any any (msg:"Testing ICMP"; sid:1000001;)
#alert udp any any -> any any (msg:"Testing UDP"; sid:1000002;)
#alert tcp any any -> any any (msg:"Testing TCP"; sid:1000003;)
#alert tcp any any -> any 80 (msg:"Possible TCP DoS"; sid:10000079 ; rev:1;)
alert tcp any any -> any 80 (msg:"MYSQL root login attempt"; sid:10000091 ; rev:1;)
alert tcp any any -> any any (content:"virus"; msg:"Virus Detected"; sid:10000082; rev:5;)
alert tcp any any -> any any (content:"https://www.youtube.com/" ; msg:"someone is using yo
```

Figure 3.6 Shows the Rules in local.rules file

Rules are also configured in myrules.rules file. It contains rules for incoming FTP connection, for detecting words like Trojan malicious activity, Terrorism content found etc. It is shown in Figure 3.7

26
Figure 3.7 Rules Configured in myrules.rules file

The rules created contains fields like, msg (message to be matched), sid (Snort identification number which is different for each rule), nocase modifier (for no case sensitivity) and two or three words can also be specified for matching (eg: insert|update).

Similarly rules are written in local.rules, web-client.rules, chat.rules and multimedia.rules. Figure 3.8 shows the screen shot for writing the rules for detecting ping flood (DDoS) attacks.

```plaintext
#--------------------
# DDoS RULES
#--------------------

alert icmp any any -> any any (msg:"ping flood attack is detected"; sid:10000098;)
drop icmp any any -> any any (itype:8; detection_filter:track by src, count 20, seconds 5; sid:100121)
alert tcp any any -> 10.1.1.100 3128 (msg:"syn flood"; flow:established,to_server; sid:1000009; rev:1;)
```

Figure 3.8 Screenshot of rules for Detecting DDoS flood attacks
Here the rule explains that it will drop all the ICMP packets containing the itype. If the Snort sees 20 pings with a span of 5 seconds then it is going to drop and then an alert is being generated.

3) Creating the attacks using Virtual Machine:

   In order to test the prototype 4 attacks are created in the virtual machine and then tested against the developed prototype.

   The detailed procedure for creating the attacks is described below.

   **SYN Flood attack:**

   In order to create a SYN flood attack, a program is written manually to flood the port 80 of the physical machine and then the program is executed in the virtual machine. Then the tcp trace present in the physical machine redirects the packets to the port 8080 of the web application server. As, Snort executes the rules, it will detect all the packets coming to port 80 and it displays an alert message that “Possible TCP DOS attempt” is performed. For each packet it receives, an alert is generated. So, as multiple packets are received at port 80, the alert is displayed multiple times. These alerts are displayed on the home.jsp of the web application.

   **Database Root Login attempt:**

   Using the other virtual machine, the database login attempt is performed by the attacker on the physical machine. So, if the MySQL is logged in from virtual machine by entering the wrong password, then immediately an alert is generated by Snort in the physical machine to alarm the administrator about the illegal access to the database. So that he can take an action against it.
Detection of Virus:

A virus program is written in the virtual machine and it is made to look like the Internet explorer, by giving the virus file an image of Internet Explorer. So, if the Internet explorer is run by any user then it automatically opens multiple tabs of different URL’s, Notepads and Word pads. Then immediately Snort captures the alert and then display it in the web application as “Virus Detected”.

Using unauthorized sites:

In the network if some unauthorized websites like pornographic sites, YouTube and Facebook are opened then a rule is written in Snort to generate an alert. If somebody opens the website and then Snort captures the alert and they are displayed in home.jsp of the web application.

4) Developing an Internal capture log server:

Generally there are many syslog servers available in the market like Kiwi Syslog Server[11], for capturing the alerts coming from Snort. But in the proposed system, a log server is developed using JAVA. This log server uses socket programming to capture all the rules and then updates all the alerts in the database. The database contains the alerts in the form a table.

5) Developing a web application:

The following are the steps to develop a web application for alarm system:

- A web application called alarm notify system is developed. For the development of web application, HTML, JSP and servlets are used.
- The administrator logs in into the alarm system and finds the report sent from apache server.
- This report is obtained from the database and is used by the Apache server. As configured in the database, the data will be shown in the tabular format.

- The administrator copies the alerts and pastes them in an excel file and saves it.

- Then this report is used for sending the email to the responsible entities.

- After sending the report the administrator can resolve the alert by pressing on the image to resolve the alert.

- Then a message is displayed showing that “Alert is updated successfully as resolved”.

- The administrator then logs out of the web application.

6) Testing the configured rules using the created attacks:

The attacks that are created are tested using the rules

- The rules are tested on Snort by manually creating 3 attacks like SYN flood attack, Database Root login attempt and virus detection. The attacks are made by the virtual machine on the physical machine.

- Here in the proposed system, some attacks are being created like MySQL attack, SYN flood attack, Ping flood attack and also detection of viruses.

- The Snort is tested against these attacks, to see if it is triggering the alerts.

- The alerts are captured by the web application and then sent to the users through email.
3.5 Unified Modelling Language:

3.5.1 Use case diagram

The use case diagram shows the actor and the functions performed by the actor. Figure 3.9 shows the use case diagram for the proposed system. Here actor is the administrator who performs many functionalities like configuring the Snort (NIDS) on the system. Then opens the web application which is an alarm system and the admin logs in into it. Then runs a command on Snort to process the rules which helps in determining the alerts. The database collects the alerts and then displays it in the web application making it easier for the administrator, to send it as an email attachment.
3.5.2 Class Diagram

Unified Modelling Language contains a class diagram, which is static in nature. The class diagram here explains the major java classes that are used in the development of the web application. The class diagram explains the design of the system by representing it in a hierarchical format and also shows the classes used in the system. The class diagram consists of the classes, attributes and the methods used in those classes.
They are divided into three boxes depending on their functionality. Figure 3.10 shows the class diagram of the proposed system.
3.5.3 Sequence Diagram

Sequence diagram is one among all the diagrams in UML. It is considered as the interaction diagram because it shows the order of occurrence of events by interacting with the other processes. The processes interact through messages following a specific order or a logical order. This diagram contains lifelines, they are shown as parallel vertical lines and the arrows represent the messages used for interaction. Figure 3.11 shows the sequence diagram of the proposed system.

![Sequence Diagram]

Figure 3.11 Sequence Diagram
3.6 Environment

The hardware and software requirements that are used in the development of the project are:

3.6.1 Eclipse Luna

Eclipse is a platform for writing programs especially in Java, in other way it is considered as an IDE for developing many applications. Eclipse is enclosed with a workspace and also contains many plug-ins, for personalizing the environment in which the application is being developed. The Eclipse contains JDT (Java Development tools) which is considered as the platform for developing new applications.

3.6.2 MySQL

This is the most widely used relational databases compared to other databases, because it is an open source software. For the development of web based applications MySQL is preferred because of its robustness.

3.6.3 Apache Tomcat server

This is an open source server developed by Apache foundation. It mainly works with the web applications containing JSP, HTML and servlets. Tomcat helps in combining JSP and web pages into one common directory for the web application. It provides add-ons to user as well as system based applications for easy deployment in various environments. The most recent version of Tomcat is Apache Tomcat 8.0 which is used in my web application.
4. Functionality of the application

These are the interfaces that are obtained in the development of the proposed system.

4.1 User Interfaces

1) The application developed in the proposed system is a web based application. The login page of the alarm system is designed using JSP. The following is the url for the login page of the web application [http://localhost:8080/AlarmSystem/Login.jsp](http://localhost:8080/AlarmSystem/Login.jsp). It is shown in Figure 4.1

![Figure 4.1 Login Page of the Alarm System](image)

2) Then the administrator logs into the web application by entering the user name and password. Then clicks on “submit”. It is shown in Figure 4.2
3) Figure 4.3 shows the command prompt window for showing the initial process of configuring the Snort. After opening the command prompt type the command `snort -w`. Then it will display the basic information about the Snort.

![Command Prompt Displaying the Environment for Snort](image-url)
4) Figure 4.4 shows the main console for initializing the Snort in the command line. This is achieved by giving the command `snort -v -i1`

![Figure 4.4 Initializing and Configuring Snort](image1.png)

5) In figure 4.5 rules are being configured inside the files with file names like `web-client.rules`, `chat.rules`, `mysql.rules`, `multimedia.rules`, `local.rules` and `myrules.rules` that are present in `c:\Snort\rules` in local drive.

![Figure 4.5 Configuring the Snort rules in local drive](image2.png)
6) Figure 4.6 shows the running of Snort with a command “snort -i3 -s -l c:\snort\log\ -c c:\snort\etc\snort.conf” to match the signatures and generate the alerts.

```
+++
Initializing rule chains...
7244 Snort rules read
6826 detection rules
150 decoder rules
268 preprocessor rules
7244 Option Chains linked into 279 Chain Headers
0 Dynamic rules
+
+++
+-----------------------------[Rule Port Counts]-----------------------------+
<table>
<thead>
<tr>
<th>src</th>
<th>tcp</th>
<th>udp</th>
<th>icmp</th>
<th>ip</th>
</tr>
</thead>
<tbody>
<tr>
<td>src</td>
<td>2232</td>
<td>19</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>dst</td>
<td>3562</td>
<td>814</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>any</td>
<td>611</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>nc</td>
<td>437</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>s+o</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
+-----------------------------[Rule Port Counts]-----------------------------+
+++
+-------------------------------[detection-filter-config]---------------------------+
| memory-cap: 1048576 bytes |
+-------------------------------[detection-filter-rules]---------------------------+
+++
+-----------------------------[rate-filter-config]-----------------------------+
```

Figure 4.6 Console Showing the Execution of rules

7) Then the execution of rules is done by the MySQL database and the Apache tomcat server. MySQL retrieves the data from the local host and stores in the form of tables. Then the Apache Tomcat executes it and displays it on the web application. It is shown in Figure 4.7 and Figure 4.8.
Figure 4.7 Interface of MySQL Database.
8) Figure 4.9 shows that, the alerts are executed and then finally displayed on the home.jsp in a tabular format. It contains the columns of alert no., alert date, alert type, alert message, alert status and resolve alert.
Figure 4.9 Displays the Alerts on home.jsp of Web Application

9) Figure 4.10 shows the excel file in which the alerts are copied from the web application and then pasted in an excel file named as alerts.xls

Figure 4.10 Shows the Excel file for Displaying the Alerts.
10) Figure 4.11 shows the interface of the web application for sending the report as an attachment. Here the administrator is sending the email by entering the recipient address, subject, content and finally attaching the file.

![Image of Send Alert Notification interface]

**Figure 4.11 Interface for Sending the Email Attachment.**
11) Figure 4.12 shows that the email has been successfully sent to the given recipient.

![Figure 4.12 Email is Sent Successfully](image)

11) The email which is sent by the administrator is displayed in the inbox of the recipient. It is shown in Figure 4.13.

![Figure 4.13 Report containing Alerts in the Gmail inbox.](image)
12) After sending the alert, the administrator can click on an image which is red in color. This displays that “Alert has been updated as resolved”. It is shown in Figure 4.14

![Figure 4.14 Alert Updated Successfully as Resolved.](image)

13) After resolving the administrator can log out of the web application and exit the Snort. It is shown in Figure 4.15

![Figure 4.15 After Resolving the alert, exit Snort](image)
5. TESTING AND EVALUATION

The proposed system is an implementation of the alarming system, making the administrator aware of the vulnerabilities existing in the system. The alarming System sends email messages to the administrator which helps him to keep track of the status of the network.

This project can be evaluated based on a few test cases like:

5.1 Test cases:

Test case 1: Testing SYN flood attack

Considering a scenario, Bob and Robert are good competitors in a company of web development. Bob is becoming rich and most popular with his work. Robert couldn’t tolerate this and so, he wants to create an attack on the Bob’s system. So, he thought of flooding his web application server by using TCP SYN flood attack.

Step 1: First step, is Robert trying to send a stream of SYN packets to a particular port which is listening on Bob’s website. He considered to send all the packets to port 80 and this results in flooding and dropping of packets, which becomes an incomplete transmission.

A program is written and executed on the virtual machine, so that it is going to flood the port 80 of the physical machine. The inputs that are given are, the IP address of the physical machine and the port number. Figure 5.1 shows the execution of the Java program that is leading to a flood attack.
Step 2:

As the program is executed, open the Snort intrusion on the physical machine and then give a command (`snort -i6 -s -l c:\snort\log\ -c c:\snort\etc\snort.conf`) to execute the rules, and then the alerts are displayed on `home.jsp`. Figure 5.2 shows the alerts that are captured for SYN flood attack.
Figure 5.2 Shows the Alerts for SYN flood attack on home.jsp

Here, the input for the program is running the java code for test flood and the output is, the alerts that are generated.

Test case 2: Testing MySQL login attempt.

Consider a scenario, in which a company consisting of secured databases to store confidential information. Henry, an employee, within the company is helping the attackers outside the organization because, they are offering him a lot of money. So, he tried to login into the MySQL database with his assigned password. But as it is a highly secured database, then a root login attempt occurs. Then, Henry will be caught by the network administrators and then he will be taken into custody.

Step 1:

Figure 5.3 shows the interface of the MySQL. Here the address of the physical machine, the username is also given. If the wrong password is entered into the interface then a popup is generated, saying that an error has occurred in logging in.
Step 2:

Figure 5.4 shows the alerts that are triggered in the physical machine after running Snort. The alerts are then captured and emailed to the respective administrators.
Test case 3: Running the virus file assuming it as a genuine one

Step 1:

A program is written in .bat format for generating virus and saved as a Internet explorer (.exe) file. Now it as a secured browser, people will be running the file, then many tabs will be opened displaying various malicious web sites. Figure 5.5 Shows the page of the internet explorer where the tabs are executed.

Figure 5.5 Screenshot of Internet Explorer showing multiple tabs
Step 2:

Figure 5.6 shows the alerts that are generated after running the virus file. The alert messages are displayed on home.jsp.

<table>
<thead>
<tr>
<th>Time</th>
<th>Alert</th>
<th>Event Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>03.05.2015</td>
<td></td>
<td>&lt;33&gt;May 03 15:01:36 kirti snort: [1:10000082:5] Virus Detected (TCP) 192.168.0.4:8870 -&gt; 74.125.227.233:80 open</td>
</tr>
<tr>
<td>03.05.2015</td>
<td></td>
<td>&lt;33&gt;May 03 15:01:36 kirti snort: [1:10000082:5] Virus Detected (TCP) 192.168.0.4:8882 -&gt; 70.32.75.233:80 open</td>
</tr>
<tr>
<td>03.05.2015</td>
<td></td>
<td>&lt;33&gt;May 03 15:01:36 kirti snort: [1:10000082:5] Virus Detected (TCP) 192.168.0.4:8880 -&gt; 70.32.75.233:80 open</td>
</tr>
<tr>
<td>03.05.2015</td>
<td></td>
<td>&lt;33&gt; May 03 15:01:35 kirti snort: [1:10000082:5] Virus Detected (TCP) 192.168.0.4:8880 -&gt; 70.32.75.233:80 open</td>
</tr>
<tr>
<td>03.05.2015</td>
<td></td>
<td>&lt;33&gt; kirti snort: [1:10000030:0] server log started on port 514 open</td>
</tr>
</tbody>
</table>

**Figure 5.6 Screenshot of the Detected Virus**

**Test case 4: Testing for using unauthorized sites**

In the network if some users are using the unauthorized sites such as YouTube, Facebook or pornographic sites. Then, if the network administrator get to know about this, he can block only those websites that are frequently opened. That is possible only with the help of writing a rule in Snort and then if any websites are opened then the administrator is alerted. Figure 5.7 Shows the Screenshot for using Youtube.com.

<table>
<thead>
<tr>
<th>Time</th>
<th>Alert</th>
<th>Event Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>03.05.2015</td>
<td></td>
<td>&lt;33&gt; May 03 18:06:33 kirti snort: [1:10000012:1] someone is using youtube (TCP) 74.125.227.232:80 -&gt; 192.168.0.4:2065 open</td>
</tr>
</tbody>
</table>

**Figure 5.7 Screenshot for Using Unauthorized Websites**
6. CONCLUSION AND FUTURE WORK

Hence a real time Monitoring, Detecting and Alarming System is developed for assessing the overall security of the system and to detect any intrusions occurring in the network. A web application is being developed with the help of JAVA, HTML and JSP, the alerts generated from Snort are collected into the MySQL database and then used by the alarm notify application to email the report to the users. A virtual network is created with two virtual machines and one physical machine, bridging is done between the three systems to make them form a single network. Attacks are created in the virtual machines and then detected by Snort in the physical machine. Later the Snort generates the alerts and these alerts are stored in the database and then used by the web application for sending an email to the users.

The future scope of the project is, extending the prototype with more number of systems and other attacks like backdoor attack, Man-in-the-Middle attack and Phishing attack can also be added. The user interface of the alarming system can be made better by adding some graphical features and also automating the whole process of detecting and sending the emails. The future enhancement of the prototype is sending the report through the telecommunication networks like SMS.
BIBLIOGRAPHY AND REFERENCES


Appendix

Class Diagram for the execution of alerts:

Class Diagram for attaching the report through email:
ui.java file:

```java
package sf.jlog514.ui;

import java.awt.BorderLayout;

public class ui extends JFrame
    implements ActionListener {

    private JFileChooser fc = new JFileChooser();
    private JTextArea log = new JTextArea(500, 200);
    private JButton follow = new JButton("Follow up");
    private log514 log514;
    private Thread t;
    private int localPort = 514;

    public void actionPerformed(String s) {
        if ("StartListener").equals(s) {
            this.log514 = new log514();
            this.log514.init_thread(this, this.localPort);
            this.t = new Thread(this.log514);
            this.t.start();
            try {
                t.sleep(5000);
                log.addLog("335 kirti snort: \[1:180000030:0 \] server log started on port 514");
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
e.printStackTrace();
            }
        }
        if ("StopListener").equalsIgnoreCase(s) {
            this.log514.stopRequest();
        }
    }
    public void addLog(string i) {
        Connection con=new Connection().getConnection();
        try{
            Statement st = con.createStatement();
        i = i.replaceAll("\\\n", "");
        i = i.replaceAll("\\t", "");
        SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss");
        //this.log.append(format.format(new Date()) + ": " + i + 
        System.out.println(format.format(new Date()) + ": " + i + "\n");
        String sql="insert into log(msg, status)values(" + format.format(new Date()) + ", "+i", 'open");
        stmt.executeUpdate(sql);
        if(i[231])
            System.out.println("Success");
    catch(Exception e)    {
        System.out.println(e.getMessage());
    }
    if (this.follow.isSelected()) {
        String s = this.log.getText();
        int pos = s.length();
        this.log.setSelection(pos);     
    }
```
```java
public void OpenFile(File pfile) throws SQLException {
    if (!pfile.canRead()) {
        addLog("Can't open the file, revise the permissions");
    }
    addLog("File Opened: " + pfile.toString());
    try {
        BufferedReader in = new BufferedReader(new FileReader(pfile));
        String str;
        while ((str = in.readLine()) != null) {
            String str1;
            this.log.append(str + "\n");
        }
        in.close();
    } catch (IOException e) {
        addLog(e.getMessage());
    }
}

@Override
public void actionPerformed(ActionEvent arg0) {
    // TODO Auto-generated method stub
}
```

Log514.java file:

```java
package sf.jlog514.work;

import java.io.IOException;

public class log514
    implements Runnable
{
    private ui theUI;
    private int localPort = 8514;
    private boolean stopRequested;
    private DatagramSocket socket;
    private Thread runThread;

    public void run()
    {
        this.runThread = Thread.currentThread();
        this.stopRequested = false;
        try {
            this.socket = new DatagramSocket(this.localPort);
            System.out.println("Service log started on port " + this.localPort);
            this.theUI.addLog("Service log started on port " + this.localPort);
        }
        catch (SocketException e) {
            System.out.println("Can't start listening: " + e.getMessage());
        }
        try {
            while (!this.stopRequested) {
                DatagramPacket dato = new DatagramPacket(new byte[2048], 2048);
                this.socket.receive(dato);
                String msg = new String(dato.getData(), 0, dato.getLength());
                this.theUI.addLog(msg);
            }
        }
    }
```
public class DConnection {
    private Connection con;
    private Statement stmt;
    private int i;
    public static void main(String args[]) {
        DConnection obj = new DConnection();
        System.out.println(""+obj.getConnection());
    }
    public Connection getConnection() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            String sqlcon = "jdbc:mysql://localhost:3306/logdetails";
            con = DriverManager.getConnection(sqlcon, "root", "root");
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return con;
    }
    public int insertQuery(String sql) {
        i=0;
        this.getConnection();
        try {
            stmt = con.createStatement();
        }
    }
}

DConnection.java file:
i=stmt.executeUpdate(sql);
this.closeConnection();

} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
return i;
}

// Update Query Process

public int updateQuery(String sql){
    int i=0;
    this.getConnection();
    try {
        stmt=con.createStatement();
        i=stmt.executeUpdate(sql);
        this.closeConnection();
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return i;
}

///// closeConnection is to be called explicitly for selectQuery() call

public ResultSet selectQuery(String sql){
    ResultSet rs=null;
    this.getConnection();
    try {
        stmt=con.createStatement();
        rs=stmt.executeQuery(sql);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return rs;
}

public void closeConnection() {
    try {
        con.close();
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
package net.code.java.mail;

import java.io.File;

@WebServlet("/SendMailAttachServlet")
@MultipartConfig(fileSizeThreshold = 1024 * 1024 * 2, // 2MB
        maxFileSize = 1024 * 1024 * 10, // 10MB
        maxRequestSize = 1024 * 1024 * 50) // 50MB
public class SendMailAttachServlet extends HttpServlet {
    private String host;
    private String port;
    private String user;
    private String pass;

    public void init() {
        // reads SMTP server setting from web.xml file
        ServletContext context = getServletContext();
        host = context.getInitParameter("host");
        port = context.getInitParameter("port");
        user = context.getInitParameter("user");
        pass = context.getInitParameter("pass");
    }

    /* handles form submission */
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
        List<File> uploadedFiles = saveUploadedFiles(request);

        String recipient = request.getParameter("recipient");
        String subject = request.getParameter("subject");
        String content = request.getParameter("content");

        String resultMessage = "";
        try {
            EmailUtility.sendEmailWithAttachment(host, port, user, pass,
                    recipient, subject, content, uploadedFiles);
            resultMessage = "The e-mail was sent successfully";
        } catch (Exception e) {
            ex.printStackTrace();
            resultMessage = "There were an error: " + e.getMessage();
        } finally {
            deleteUploadFiles(uploadedFiles);
            request.setAttribute("message", resultMessage);
            request.getRequestDispatcher("/EmailForm.jsp").forward(
                    request, response);
        }
    }

    /* Saves files uploaded from the client and return a list of these files 
     * which will be attached to the e-mail message.
     */
    private List<File> saveUploadedFiles(HttpServletRequest request) 
            throws IOException, ServletException {
        List<File> listFiles = new ArrayList<File>();
        byte[] buffer = new byte[4096];
        int bytesRead = -1;
        Collection<Part> multipart = request.getParts();
        if (multipart.size() > 0) {
            for (Part part : request.getParts()) {
                // creates a file to be saved
                String fileName = extractFileName(part);
                if (fileName == null || fileName.equals("")) {
                    // not attachment part, continue
                    continue;
                }

                try {
                    // create file
                    File file = new File(fileName);
                    file.createNewFile();
                    // copy file
                    InputStream in = part.getInputStream();
                    OutputStream out = new FileOutputStream(file);
                    int c;
                    while ((c = in.read(buffer)) >= 0)
                        out.write(buffer, 0, c);
                    out.close();
                    in.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                listFiles.add(file);
            }
        }
        return listFiles;
    }
}
File saveFile = new File(fileName);
System.out.println("saveFile: " + saveFile.getAbsolutePath());
FileOutputStream outputStream = new FileOutputStream(saveFile);

// saves uploaded file
InputStream inputStream = part.getInputStream();
while ((byteRead = inputStream.read(buffer)) != -1) {
    outputStream.write(buffer, 0, byteRead);
}
outputStream.close();
inputStream.close();
listFiles.add(saveFile);
}
return listFiles;
}

/**
 * Retrieves file name of a upload part from its HTTP header
 */
private String extractFileName(Part part) {
    String contentDisp = part.getHeader("content-disposition");
    String[] items = contentDisp.split(";");
    for (String s : items) {
        if (s.trim().startsWith("filename")) {
            return s.substring(s.indexOf(\"=") + 2, s.length() - 1);
        }
    }
    return null;
}

/**
 * Deletes all uploaded files, should be called after the e-mail was sent.
 */
private void deleteUploadFiles(List<File> listFiles) {
    if (listFiles != null && listFiles.size() > 0) {
        for (File aFile : listFiles) {
            aFile.delete();
        }
    }
}
package net.codejava.mail;

import java.io.File;

/**
 * A utility class for sending e-mail message with attachment.
 * @author www.codejava.net
 */

public class EmailUtility {

    public static void sendEmailWithAttachment(String host, String port,
            final String userName, final String password, String toAddress,
            String subject, String message, List<File> attachedFiles)
            throws AddressException, MessagingException {

        // sets SMTP server properties
        Properties properties = new Properties();
        properties.put("mail.smtp.host", host);
        properties.put("mail.smtp.port", port);
        properties.put("mail.smtp.auth", "true");
        properties.put("mail.smtp.starttls.enable", "true");
        properties.put("mail.user", userName);
        properties.put("mail.password", password);

        // creates a new session with an authenticator
        Authenticator auth = new Authenticator() {
            public PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(userName, password);
            }
        };
        Session session = Session.getInstance(properties, auth);

        // creates a new e-mail message
        Message msg = new MimeMessage(session);
        msg.setFrom(new InternetAddress(userName));
        InternetAddress[] toAddresses = { new InternetAddress(toAddress) };
        msg.setRecipients(Message.RecipientType.TO, toAddresses);
        msg.setSubject(subject);
        msg.setSentDate(new Date());

        // creates message part
        Multipart messageBodyPart = new MimeBodyPart();
        messageBodyPart.setContent(message, "text/html");

        // creates multi-part
        Multipart multipart = new MimeMultipart();
        multipart.addBodyPart(messageBodyPart);

        // adds attachments
        if (attachedFiles != null && attachedFiles.size() > 0) {
            for (File aFile : attachedFiles) {
                try {
                    MimeBodyPart attachPart = new MimeBodyPart();
                    attachPart.attachFile(aFile);
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
        }

        // sets the multi-part as e-mail's content
        msg.setContent(multipart);

        // sends the e-mail
        Transport.send(msg);
    }
}