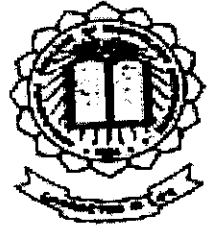


R-3308



# IMPROVING INTRUSION DETECTION SYSTEM FALSE ALARM RATIO USING HONEY POT

A PROJECT REPORT

*Submitted by*

**DHIVYA.S**

**71206205011**

**SARIGHA.R**

**71206205042**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**INFORMATION TECHNOLOGY**

**KUMARAGURU COLLEGE OF TECHNOLOGY**

**COIMBATORE-641 006**

**ANNA UNIVERSITY: CHENNAI 600 025**

**April 2010**

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

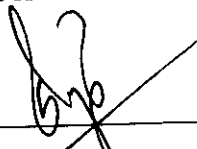
Certified that this project report entitled “ **Improving Intrusion Detection System False Alarm Ratio Using Honeypot** ” is the bonafide work of

**DHIVYA.S** 71206205011

**SARIGHA.R** 71206205042

who carried out the project work under my supervision.

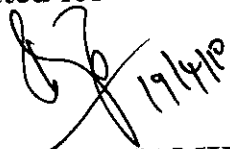
  
SIGNATURE


  
SIGNATURE

**Ms.N.Rajathi.,M.E.,**  
**Assistant Professor,**  
**Department of**  
**Information Technology,**  
**Kumaraguru College of Technology,**  
**Chinnavedampatti Post,**  
**Coimbatore - 641006.**

**Dr.L.S.Jayashree, Ph.D.**  
**Professor & Head,**  
**Department of**  
**Information Technology,**  
**Kumaraguru College of Technology ,**  
**Chinnavedampatti Post,**  
**Coimbatore - 641006**

Submitted for viva-voice examination held on 19-04-2010 .

  
INTERNAL EXAMINAR

  
EXTERNAL EXAMINAR

## DECLARATION

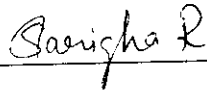
We hereby declare that the project entitled “ **IMPROVING INTRUSION DETECTION SYSTEM FALSE ALARM RATIO USING HONEYPOT**” submitted in partial fulfillment to Anna University as a project work of Bachelor of Technology ( Information Technology ) degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru College of Technology , Coimbatore.

Place: Coimbatore

Date: 19-04-2010

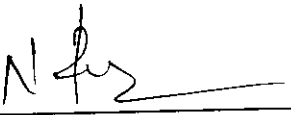


[S.DHIVYA]



[R.SARIGHA]

Project Guided by



[Ms.N.Rajathi, M.E., ]



[Dr.L.S.Jayashree, M.E.,Ph.D.]

## **ACKNOWLEDGEMENT**

We express our sincere thanks to our Chairman **Padmabhushan Arutselvar Dr.N.Mahalingam B.Sc., F.I.E.**, and correspondent **Shri.M.Balasubramaniam,M.Com.,M.B.A.**, Director **Dr.J.S.Shanmugam, Ph.D** for all their support and ray of strengthening hope extended. We are immensely grateful to our Principal, **Dr.S.Ramachandran ,Ph.D**, for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy, BE (HONS), Ph.D.**, Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our project co-coordinator **Dr.L.S.Jayashree,M.E.,Ph.D**, HOD, Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide **Ms.N.Rajathi,M.E.**, Assistant Professor, Department of Information Technology for her helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project. We also thank all our friends who helped us to complete this project successfully.

**ABSTRACT**

---

## **ABSTRACT**

Traditional firewall and intrusion detection systems (IDS) are used to detect possible attacks from the network, they often make wrong decisions and block the legitimate connections and propose a new architecture which is composed of distributed agents and honeypot. The main focus of our approach lies in reducing the false alarm rate of the attack detection. Using the honeypot scheme, this system is able to avoid many wrong decisions made by IDS. In this system alarming adversaries, initially detected by the IDS, will be rerouted to a honeypot network for a more close investigation. If as a result of this investigation, it is found that the alarm decision made by the IDS of the agent is wrong, the connection will be guided to the original destination in order to continue the previous interaction. This action is hidden to the user. Such a scheme significantly decreases the alarm rate and provides a higher performance of IDS. In this proposed architecture, a theoretical analysis of its behavior is given and its possible extension and implementation are explained.

## CONTENTS

---

## TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	Abstract	iii
	List of Figures	vii
	List of Abbreviations	viii
<b>1.</b>	<b>Introduction</b>	
	1.1 Introduction to IDS	2
	1.2 Introduction to Honeypot	3
	1.3 Objective	4
<b>2.</b>	<b>Literature Review</b>	
	2.1 Intrusion Detection System	6
	2.2 Major Network Attacks	
	2.2.1 Denial of Service	8
	2.2.2 IP Spoofing	10
	2.2.3 NIMDA Attack	12
	2.3 Existing System	15
	2.4 Proposed System	16



<b>3.</b>	<b>System Requirements</b>	
	3.1 Hardware Requirements	18
	3.2 Software Requirements	18
	3.3 Software Description	
	3.3.1 Java	18
	3.3.2 Java Platform	22
<b>4.</b>	<b>Details of Methodology Employed</b>	
	4.1 Input Design	24
	4.2 System Flow Diagram	25
<b>5.</b>	<b>Performance Evaluation</b>	32
<b>6.</b>	<b>Conclusion</b>	34
<b>7.</b>	<b>Future Work</b>	36
<b>8.</b>	<b>Appendix</b>	
	8.1 Sample Source Code	38
	8.2 Screen Shots	59
<b>9.</b>	<b>References</b>	66

## LIST OF FIGURES

<b>S.NO</b>	<b>TITLE</b>	<b>FIGURE NO</b>	<b>PAGE NO</b>
<b>1</b>	<b>System Flow Diagram</b>	<b>1</b>	<b>25</b>
<b>2</b>	<b>System Architecture</b>	<b>2</b>	<b>26</b>

## LIST OF ABBREVIATIONS

<b>TCP</b>	:	Transport Control Protocol
<b>IP</b>	:	Internet Protocol
<b>JDBC</b>	:	Java database Connectivity
<b>UDP</b>	:	User Datagram Protocol
<b>RMI</b>	:	Remote Method Invocation
<b>CORBA</b>	:	Common Object Request Broker Architecture
<b>HTTP</b>	:	Hypertext Transfer Protocol Secure
<b>GIDO</b>	:	Generalised Intrusion Detection Object
<b>IMA</b>	:	Investigative Mobile Agents



# 1.INTRODUCTION

## 1.1 INTRODUCTION TO IDS

By increasing the usage of the Internet and implementing commonly used tasks through it, the concept of distributed applications has been considerably grown. Currently firewall and Intrusion Detection Systems (IDS) have been practically developed to block variety of threats through incoming port connections. Intrusion detection technology in general helps to find out the illegal intrusions from inside and outside of the local network by tracking the intruders' trail, such as the records of failure access trails. A typical IDS consists of the following parts: event generator, event analyzer, response units and event databases. The data are exchanged in form of GIDOs (Generalized Intrusion Detection Objects) which are represented by a standard common format CIDF between the parts. GIDO is the specification of messaging as its encoded content is either some particular occurrence happened at some particular time, or some conclusion about a set of events or an intrusion to carry out an action. Each part might be implemented as a single process on a computer (or even a thread), or might be a collection of many processes on a number of computers.

The event generator obtains events from data outside the intrusion detection system, generates events based on the traffic thereon and handovers them in GIDO format to the other parts. Event generators provide events as soon as they occur with the possible exception of transport queuing and based on some predefined rules which are dynamically updated. Storage of events is handled in related event databases. When the event generator obtains the required information, it sends an analysis request to its relevant event analyzer. The packets in the monitored network are compared against a repository of signatures which define characteristics of the intrusion while event analyzer analyzes the data and generates new GIDOs which presumably represent some kind of

## 1.2 INTRODUCTION TO HONEYPOT

The honeypot is a trap set to detect, deflect, or in some manner counteract attempts at unauthorized use of information systems. Generally it consists of a computer, data, or a network site that appears to be part of a network, but is actually isolated, (un)protected, and monitored, and which seems to contain information or a resource of value to attackers.

A honeypot is valuable as a surveillance and early-warning tool. While it is often a computer, a honeypot can take other forms, such as files or data records, or even unused IP address space. A honeypot that masquerades as an open proxy to monitor and record those using the system is a sugarcane. Honeypots should have no production value, and hence should not see any legitimate traffic or activity. Whatever they capture can then be surmised as malicious or unauthorized. Honeypots can carry risks to a network, and must be handled with care. If they are not properly walled off, an attacker can use them to break into a system.

Victim hosts are an active network counter-intrusion tool. These computers run special software, designed to appear to an intruder as being important and worth looking into. In reality, these programs are dummies, and their patterns are constructed specifically to foster interest in attackers. The software installed on, and run by, victim hosts is dual purpose. First, these dummy programs keep a network intruder occupied looking for valuable information where none exists, effectively convincing him or her to isolate themselves in what is truly an unimportant part of the network. This decoy strategy is designed to keep an intruder from getting bored and heading into truly security-critical systems. The second part of the victim host strategy is intelligence gathering. Once an intruder has broken into the victim host, the machine or a network administrator can examine the intrusion methods used by

the intruder. This intelligence can be used to build specific countermeasures to intrusion techniques, making truly important systems on the network less vulnerable to intrusion.

### **1.3 OBJECTIVE**

The main aim of the proposed architecture is to decrease false positives of the network. There are two types of errors concerns, TypeI and TypeII. TypeI error named false positive is the error of rejecting a null hypothesis when it is actually true, in other word, it is the error of rejecting a hypothesis that should have been accepted. TypeII error named false negative is the error of accepting a null hypothesis that should have been rejected. In this paper we cope with false positive errors.

The main focus of our approach lies in reducing the false alarm rate of the attack detection. Using the honeypot scheme, this system is able to avoid many wrong decisions made by IDS. In this system alarming adversaries, initially detected by the IDS, will be rerouted to a honeypot network for a more close investigation. If as a result of this investigation, it is found that the alarm decision made by the IDS of the agent is wrong, the connection will be guided to the original destination in order to continue the previous interaction. This action is hidden to the user. Such a scheme significantly decreases the alarm rate and provides a higher performance of IDS.





## **2. LITERATURE REVIEW**

### **2.1 INTRUSION DETECTION SYSTEM**

#### **COMPONENTS OF THE IDS IN A SUBNET**

The fundamental design of our proposed hybrid model in each subnet consists of four main components namely IDS Control Center, Agency, Static Agent Detectors, and Specialized Investigative Mobile Agent Detectors.

#### **Agency**

Mobile Agents need an environment to become alive. That environment is called Agency. An agency is responsible for hosting and executing agents in parallel and provides them with environment so that they can access services, communicate with each other, and migrate to other agencies. An agency also controls the execution of agents and protects the underlying hardware from unauthorized access by malicious agents.

#### **Static Agent Detectors**

Static Agent Detectors (SAD) act like host monitors, generating ID events whenever traces of an attack is detected, and these events are sent in the form of structured messages to IDS Control Center . For example, when SAD identifies failed password guessing attempts as a suspicious activity, an ID event is generated to check for corresponding attack. SAD is capable of monitoring the host for different classes of attacks. The SAD is responsible for parsing the log files, checking for intrusion related data pattern in log files, separating data related to the attack from the rest of the data and formatting the data as required by the investigative MA. The architecture of our IDS allows applying components of other project as an intrusion detection sensor. In that case static agent detectors will work on top of those sensors. For instance the SNORT network intrusion detection system and its sensors can be used to do packet

## **Investigative Mobile Agents**

Investigative Mobile Agents (IMA) are responsible for collecting evidences of an attack from all the attacked hosts for further analysis. Then, they have to correlate and aggregate that data to detect distributed attacks. Each IMA is only responsible for detecting certain types of intrusions. This makes it easier for updating when new types of intrusion is found or new types of detection method is invented. In addition, it lets Mobile agents carry less data and code. In addition, the IDS can also be updated and extended by adding new MAs. The investigative MA uses List of Compromised Agency (LCA) to identify its itinerary for visiting Hosts.

## **IDS Control Center**

An Intrusion Detection System Control Center is a central point of IDS components administration in each subnet. It includes following components:

- **Databases:** There should be a database of all intrusion patterns which can be used by Alerting Console to raise the alarm if patterns matched with the detected suspicious activities. All events IDs which reported by SA are stored in another database. In addition all related system logs should be stored in a database as well.
- **Alerting Console:** This component compare the spotted suspicious activity with intrusions' database and raise the alarm if they are matched.
- **Agent generator:** Functions to generate task specific agent for detecting intrusions even new ones by using knowledge that is generated by data mining inference engine obtained from previous experience.

- **Mobile agent dispatcher:** Dispatched investigative mobile agents to the host based on the ID of event or suspicious determines list of compromised Agencies (LCA) for investigative MAs.
- **Data mining inference engine:** Uses machine learning to deduce knowledge to detect new intrusions from System databases which contains detected intrusion and system logs and coming information from SAs.
- **Trust level manager:** Defines trust level for all agencies in the subnet, furthermore it keep the trust level of the other IDS control centers in the same neighborhood of network

## 2.2 MAJOR NETWORK ATTACKS

### 2.2.1 DENIAL OF SERVICE

Denial of Services Attacks (DOS) is a constant danger to web sites. DOS has received increased attention as it can lead to a severe lost of revenue if a site is taken offline for a substantial amount of time. There are many types of denial of service attacks but two of the most common are Ping of Death and TCP SYN Flood. We have chosen to implement these two techniques and add Distributed DOS (DDOS) as well. In a Ping of Death attack, a host sends hundreds of ping requests (ICMP Echo Requests) with a large or illegal packet size to another host in attempt to knock it offline or to keep it so busy responding with ICMP Echo replies that it cannot service its clients.

A TCP SYN Flood attack takes advantage of the standard TCP three-way handshake by sending a request for connection with an invalid return address. In this paper we demonstrate DDOS by creating a worm like program that installs programs on remote machines to attack a particular server. These attackers listen

in the background for a message from a master program that will tell these attackers to launch a DOS attack against a machine.

DDOS attacks are difficult to stop because they can be coming from anywhere in the world. We will implement a DDOS attack by launching the Ping of Death implementation against a victim computer from several other workstations.

## **DISTRIBUTED DENIAL OF SERVICE WITH PING OF DEATH PAYLOAD IMPLEMENTATION**

To implement DDOS, a worm like program is created to simulate self-propagation onto many hosts on a network. However, creating an actual worm is beyond therefore, we used a small Java program to simulate such a worm. Though it carries the payload and waits to receive orders from a master program, the worm does not self propagate. We simply placed the application on each host machine manually for simulation purposes. The worm-like zombie program will launch a Ping of Death attack from multiple hosts coordinated by a master program. The applications handle all communication between each other. When the master program orders the attack, a message is sent to all the zombies that makes them release their Ping of Death payload against a victim host that is specified by the master program.

The Java implementation has been built using TCP sockets and serializable Java objects. Serializable Java objects can be transferred to remote servers and then executed with all of its information intact. The serializable Java objects have all the instructions needed to launch a particular type of attack. When a user wishes to initiate an attack, he or she starts up the master program and specifies which server to attack. The master program then looks up the IP addresses of all known zombie programs and what ports they are listening on by accessing a

attack type specified, and sends it to every zombie listed in its configuration file over a TCP socket.

The zombie program recognizes that it has received a message and reads from a TCP socket the serializable java object. It then deserializes it and executes it, which in turn will launch the DOS attack.

Most of mobile agent based intrusion detection systems, such as Autonomous Agents for Intrusion Detection (AAFID) follow a hierarchical structure. In this type of IDS, data is collected both at the host and network levels by reviewing audit trails and or monitoring packets in a network. All collected data are sent to the central data coordinator. The coordinator then analyzes all the information received to decide the status of the system, and actions that need to be taken.

The mentioned system can perform intrusion detection task, however it is vulnerable. If any part of the internal nodes (or even the root node) is disabled, the functioning of that of branch of IDS will be disqualified. In addition they are not flexible, and not completely distributed. Furthermore, in those hierarchical models no approach has been taken to respond to attack against intrusion detection system itself. As a result, if intruder can gain access to IDS coordinator (IDS control center), the whole system will be affected. The performance of IDS with mobile agents is considerably relying on the produced network load by the agents.

### **2.2.2 IP SPOOFING**

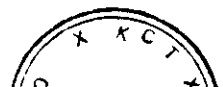
IP spoofing refers to the creation of Internet Protocol (IP) packets with a forged source IP address, called spoofing, with the purpose of concealing the identity of the sender or impersonating another computing system.

The basic protocol for sending data over the Internet network and many other computer networks is the Internet Protocol ("IP"). The header of each IP packet contains, among other things, the numerical source and destination address of the packet. The source address is normally the address that the packet was sent from. By forging the header so it contains a different address, an attacker can make it appear that the packet was sent by a different machine. The machine that receives spoofed packets will send response back to the forged source address, which means that this technique is mainly used when the attacker does not care about the response or the attacker has some way of guessing the responses. In certain cases, it might be possible for the attacker to see or redirect the response to his own machine. The most usual case is when the attacker is spoofing an address on the same LAN or WAN. Hence the attackers have an unauthorized access over computers.

IP spoofing can also be a method of attack used by network intruders to defeat network security measures, such as authentication based on IP addresses. This method of attack on a remote system can be extremely difficult, as it involves modifying thousands of packets at a time. This type of attack is most effective where trust relationships exist between machines. For example, it is common on some corporate networks to have internal systems trust each other, so that users can log in without a username or password provided they are connecting from another machine on the internal network (and so must already be logged in). By spoofing a connection from a trusted machine, an attacker may be able to access the target machine without an authentication.

Configuration and services that is vulnerable to IP spoofing are

- ❖ RPC (Remote Procedure Call services)
- ❖ Any service that uses IP address authentication



Packet filtering is one defense against IP spoofing attacks. The gateway to a network usually performs ingress filtering, which is blocking of packets from outside the network with a source address inside the network. This prevents an outside attacker spoofing the address of an internal machine. Ideally the gateway would also perform egress filtering on outgoing packets, which is blocking of packets from inside the network with a source address that is not inside. This prevents an attacker within the network performing filtering from launching IP spoofing attacks against external machines. It is also recommended to design network protocols and services so that they do not rely on the IP source address for authentication.

### **2.2.3 NIMDA ATTACK**

Nimda is a computer worm, and is also a file infector. It quickly spread, eclipsing the economic damage caused by past outbreaks such as Code Red. Multiple propagation vectors allowed Nimda to become the Internet's most widespread virus/worm within 22 minutes. The worm was released on September 18, 2001. Due to the release date, exactly 1 week after the attacks on the World Trade Center and Pentagon, some media quickly began speculating a link between the virus and Al Qaeda, though this theory ended up proving unfounded.

Nimda affected both user workstations (clients) running Windows 95, 98, Me, NT, 2000 or XP and servers running Windows NT and 2000. The worm's name spelled backwards is "admin".

## Methods of Infection

Nimda was so effective partially because it—unlike other infamous malware like the Morris worm or Code Red—uses five different infection vectors:

- ❖ Via email
- ❖ Via open network shares
- ❖ Via browsing of compromised web sites
- ❖ Exploitation of various Microsoft IIS 4.0/5.0 directory traversal vulnerabilities.
- ❖ Via back doors left behind by the "Code Red II"

Nimda's payload appears to be the traffic slowdown itself - that is, it does not appear to destroy files or cause harm other than the considerable time that may be lost to the slowing or loss of traffic known as denial-of-service and the restoring of infected systems. With its multi-pronged attack, Nimda appears to be the most troublesome virus of its type that has yet appeared. Its name (backwards for "admin") apparently refers to an "admin.DLL" file that, when run, continues to propagate the virus.

To briefly summarize what Nimda does:

- ❖ It probes each IP address within a randomly-selected range of IP addresses, attempting to exploit weaknesses that, unless already patched, are known to exist in computers with Microsoft's Internet Information Server. A system with an exposed IIS Web server will read a Web page containing an embedded JavaScript that automatically executes, causing the same JavaScript code to propagate to all Web pages on that server.
- ❖ As people (those with Microsoft Internet Explorer browsers at the 5.01 or



download pages with the JavaScript that automatically executes, causing the virus to be sent to other computers on the Internet in a somewhat random fashion.

- ❖ Nimda also can infect users within the Web server's own internal network that have been given a network share (a portion of file space).
- ❖ Finally, one of the things that Nimda has an infected system do is to send an e-mail with a "readme.exe" attachment to the addresses in the local Windows address book. A user who opens or previews this attachment (which is a Web page with the JavaScript) propagates the virus further.

To summarize the preventive action:

- ❖ Server administrators should get and apply the cumulative IIS patch that Microsoft has provided for previous viruses and ensure that no one at the server opens e-mail.
- ❖ PC users should never open a "readme.exe" attachment sent by e-mail. They should also update their Internet Explorer version to IE 5.5 SP2 or IE 6.0.

## 2.3 EXISTING SYSTEM

Intrusion detection system helps to find out the illegal intrusions from inside and outside of the local network by tracking the intruders' trail, such as the records of failure access trails. A typical IDS consists of the following parts: event generator, event analyzer, response units and event databases. The event generator obtains events from data outside the intrusion detection system, generates events based on the traffic thereon and handovers them in GIDO format to the other parts. Event generators provide events as soon as they occur with the possible exception of transport queuing and based on some predefined rules which are dynamically updated. Storage of events is handled in related event databases. When the event generator obtains the required information, it sends an analysis request to its relevant event analyzer. The packets in the monitored network are compared against a repository of signatures which define characteristics of the intrusion while event analyzer analyzes the data and generates new GIDOs which presumably represent some kind of synthesis or summary of the input events.

We use honey pot together with IDS in order to observe the suspected attacker's activities in more details. The reason for this combination is that honey pot is able to log the files and analyze them but it will be overloaded by the normal traffic analysis. However IDS is used for attack detection but it generates lots of false alarms which are not really attacks

## 2.4 PROPOSED SYSTEM

Outline the requirements of the proposed infrastructure which is considered as a solution to decrease false positives of the network . There are two types of errors concerns, TypeI and TypeII. TypeI error named false positive is the error of rejecting a null hypothesis when it is actually true, in other word, it is the error of rejecting a hypothesis that should have been accepted. TypeII error named false negative is the error of accepting a null hypothesis that should have been rejected. The proposed work dealt with false positive errors.

This topology consists of distributed agents (clients or servers), each with associated with an IDS. The architecture can be software resides on an individual computer or a set of computers. The Honey Pot, as a main part of the topology, cooperates with honey pot when a suspected threat is detected through the IDS of one of the agents. As a matter of fact the manager provides an attractive but diversionary playground for the suspected hacker in which honey pot interacts and obtains the required information from the end user.

A new system on a distributed agent based network. The system, in case of suspended adversaries, changes the path of connection to the honey pot system for a more close investigation. Decreasing false alarms rate is the system's objective. The system also tries to store any necessary information about the identified attacker. In the simulated system we obtained good results which were declaring the decreasing in false positive ratio. Due to the filtering of IDS the load of the system directed to the Honey pot is decreased leading the Honey pot to better performance in packet dropping



### **3. SYSTEM REQUIREMENTS**

A complete specification of hardware and software requirements is essential for the success of software development. This software has been developed with very powerful and high performance multi-user computing system. It is applicable in the areas where much processing speed is required.

#### **3.1 HARDWARE REQUIREMENTS**

Processor	:	Intel Pentium IV
Processor Speed	:	1.4 GHz
Memory (RAM)	:	512MB
Hard disk	:	40GB
Monitor	:	14 "IBM color monitor
Input Device	:	Keyboard (104)

#### **3.2 SOFTWARE REQUIREMENTS**

Operating System	:	Windows XP
Language	:	JAVA
Development Kit	:	JDK1.5 and above
Platform	:	Independent

#### **3.3 SOFTWARE DESCRIPTION**

##### **3.3.1 JAVA**

Java is a new computer programming language developed by Sun Microsystems. Java has a good chance to be the first really successful new computer language in several decades. Advanced programmers like it because it has a clean, well-designed definition. Business likes it because it dominates an

Java has several important features:

- ❖ A Java program runs exactly the same way on all computers. Most other languages allow small differences in interpretation of the standards.
- ❖ It is not just the source that is portable. A Java program is a stream of bytes that can be run on any machine. An interpreter program is built into Web browsers, though it can run separately. Java programs can be distributed through the Web to any client computer.
- ❖ Java applets are safe. The interpreter program does not allow Java code loaded from the network to access local disk files, other machines on the local network, or local databases. The code can display information on the screen and communicate back to the server from which it was loaded.

## **FEATURES**

### **Java is simple:**

The most important simplification is that Java does not use pointers and implements automatic garbage collection so that we don't need to worry about dangling pointers, invalid pointer references, and memory leaks and memory management.

### **Java is object-oriented:**

This means that the programmer can focus on the data in his application and the interface to it. In Java, everything must be done via method invocation for a Java object.

### **Java is distributed:**

Java is designed to support applications on networks. Java supports various levels of network connectivity through classes in `java.net`. For instance, the `URL` class provides a very simple interface to networking.

### **Java is robust:**

Java is designed for writing highly reliable or robust software. Java puts a lot of emphasis on early checking for possible problems, later dynamic checking, and eliminating situations that are error prone. The removal of pointers eliminates the possibility of overwriting memory and corrupting data.

### **Java is secure:**

Java is intended to be used in networked environments. Toward that end, Java implements several security mechanisms to protect us against malicious code that might try to invade your file system. Java provides a firewall between a networked application and our computer.

### **Java is architecture-neutral:**

Java programs are compiled to an architecture neutral byte-code format. The primary advantage of this approach is that it allows a Java application to run on any system that implements the Java Virtual Machine.

### **Java is portable:**

The portability actually comes from architecture-neutrality. But Java goes even further by explicitly specifying the size of each of the primitive data types to eliminate implementation-dependence.

### **Java is interpreted:**

The Java compiler generates byte-codes. The Java interpreter executes the translated byte codes directly on system that implements the Java Virtual Machine

### **Java is high-performance:**

Compared to those high-level, fully interpreted scripting languages, Java is high-performance. If the just-in-time compilers are used, Sun claims that the performance of byte-codes converted to machine code are nearly as good as native C or C++.

### **Java is multithreaded:**

Java provides support for multiple threads of execution that can handle different tasks with a Thread class in the java.lang Package. The thread class supports methods to start a thread, run a thread, stop a thread, and check on the status of a thread.

### **Java is dynamic:**

Java loads in classes, as they are needed, even from across a network. This makes an upgrade to software much easier and effectively. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed.



## 3.2 JAVA Platform

A platform is the hardware or software environment in which a program runs. Most Platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's software –only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- ❖ The Java Virtual Machine (JVM),
- ❖ The Java Application Programming Interfaces (Java API),

The JVM has been explained above. It's the base for the Java platform and is ported onto various hardware-based platforms. The Java API is a large collection of ready-made software components that provide many useful capabilities, such as GUI. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages.

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.



## **4. DETAILS OF METHODOLOGY EMPLOYED**

### **System Design**

System design is a solution to the creation of a new system. It provides the understanding and procedural details necessary for implementing the system. The emphasis is on translating the performance requirements into design specification. Design goes through logical and physical stages of the development.

#### **4.1 INPUT DESIGN**

Input Design is one of the most important phase system designs. Input design is the process where. The Input received in the system are planned and designed, so as to get only necessary information from the user, eliminating the information that is not required. The aim of the input design is to ensure the maximum possible levels of accuracy and also ensures that the input is accessible that is understood by the user.

The input design is the part of overall system design, which requires very careful attention. If the data going into the system is incorrect then the processing and output will magnify the errors.

The objectives considered during input designation

- Nature of validation.
- Flexibility of validation rules
- Handling of properties within the input documents
- Screen design to ensure accuracy and efficiency of input relationship with files

## 4.2 SYSTEM FLOW DIAGRAM

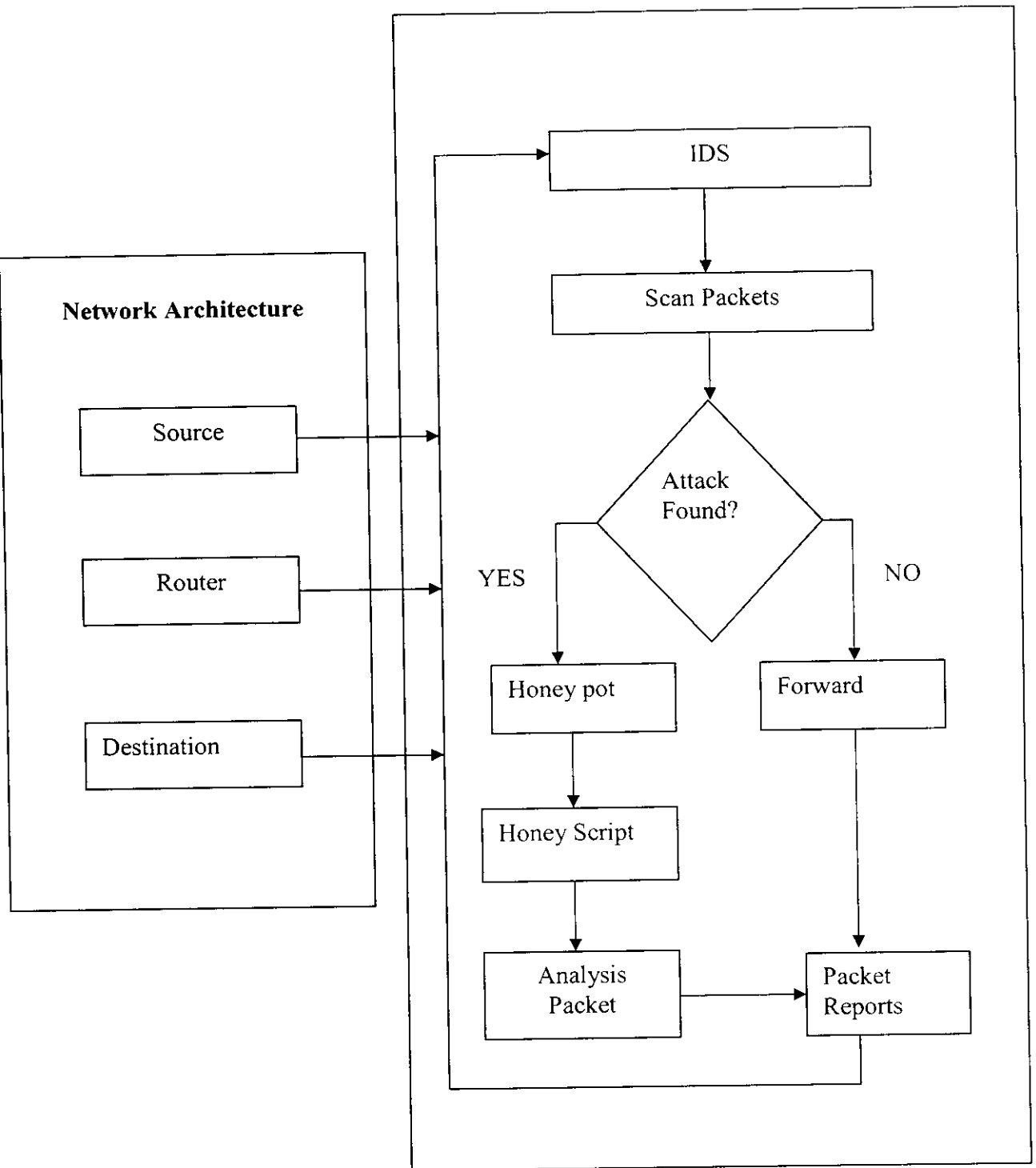
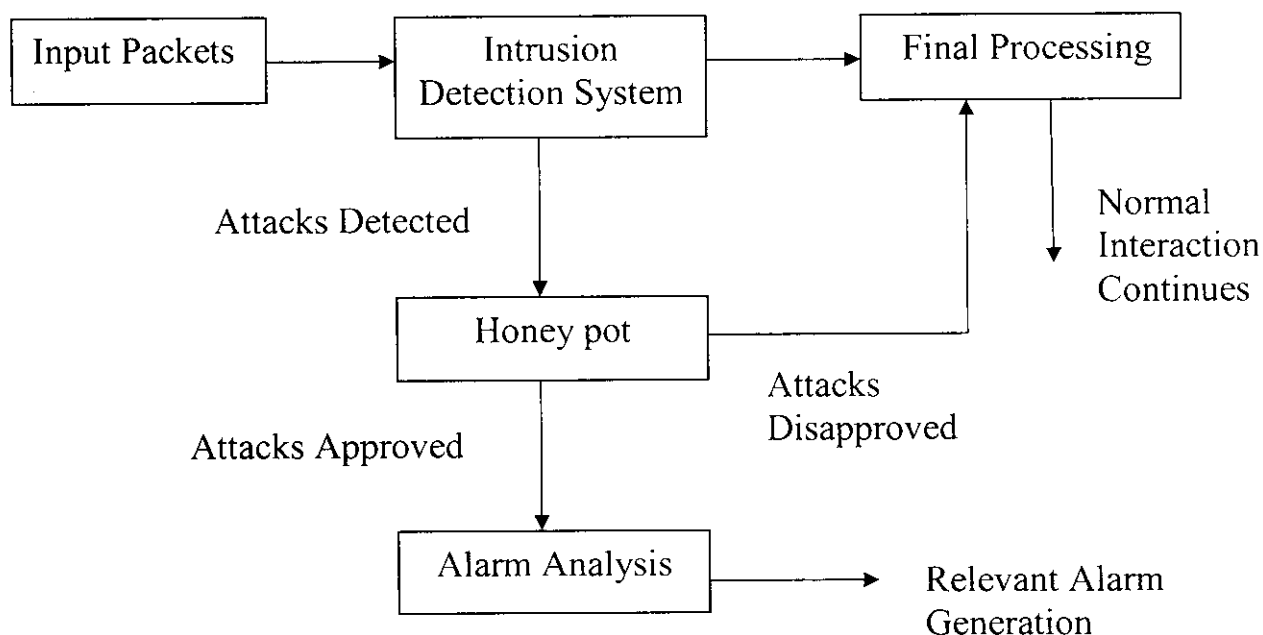


Figure 1 System Flow Diagram

The topology consists of distributed agents (clients or servers), each with associated with an IDS. The architecture can be software resides on an individual computer or a set of computers.

The honey pot when a suspected threat is detected through the IDS of one of the agents. As a matter of fact an attractive but diversionary playground for the suspected hacker in which honey pot interacts and obtains the required information from the end user. In this configuration, main network is a "TCP/IP based network" which contains clients and servers. Honey Pot is a honey pot system that is configured within the system. We assume a "switched network" in our approach, with a configurable switch which is supposed to mirror all the traffic to the specific port; its duty is to send a copy of all packets to the port of the honey pot .



**Figure 2. System Architecture**

A variety of servers and services can be supported by this system such as application, authentication and authorization, database servers, firewalls, gateways and workstations. Figure 2 illustrates the main structure's functionality.

## **Intrusion Detection System**

The main task of intrusion detection systems is defense of a computer system by detecting an attack and possibly repelling it. Detecting hostile attacks depends on the number and type of appropriate actions. Diverting the intruders attention from protected resources is another task. Both the real system and a possible trap system are constantly monitored. Data generated by intrusion detection systems is carefully examined (this is the main task of each IDS) for detection of possible attacks (intrusions).

Once an intrusion has been detected, IDS issues alerts notifying administrators of this fact. The next step is undertaken by the Honey pot. Among various IDS tasks, intruder identification is one of the fundamental ones. It can be useful in the forensic research of incidents and installing appropriate patches to enable the detection of future attack attempts targeted on specific persons or resources.

Intrusion detection may sometimes produce false alarms, for example as a result of malfunctioning network interface or sending attack description or signatures via email. Hence the attack is directed to the honey pot to improve the accuracy of the system.

## **Honey Pot**

As a virtual unreal network, Honey Pot has been used in this topology. Honey Pot is generally designed for networks. Its duty is to detect unauthorized activities by monitoring all the unused IPs in the network. It is capable of controlling all the unused IPs at the same time. Any time Honey Pot generates an alert, it is most likely a real attack not a false alarm, thus the director approves the attack. Honey Pot can also detect unknown attacks such as new RPC 0-day as well. With Honey Pot, we have additional option of creating emulated services that interact with the attacker. The emulated services allow the director to determine what the attacker is attempting to do, what he is looking for. This is done by creating scripts that listen to specific ports and then interact with the attacker in a predetermined manner. We have additional option of creating emulated services that interact with the attacker. The emulated services allow the director to determine what the attacker is attempting to do, what he is looking for. This is done by creating scripts that listen to specific ports and then interact with the attacker in a predetermined manner. It continuously monitors the unused IP space. Honey Pot is only able to interact with attackers.

## **Records**

The records provide a database including a crime table, policy table, reports and event logs. The honey pot for making decisions refers to this database and also enters new rules and signatures derived from new attack. The honey pot constantly modifies the signature lists that are used by external agents. Records part is equipped with a system which organizes the signatures coming through the database and enables to get access to the current signatures and update them or insert new ones. Therefore all the important attacks are lead to Honey Pot for a more close investigation.

## **External Agents**

The first step of the detection process is done via agents, the real intrusion detection systems used for the primary detection. Considerable point in this system is that the agent handles all the network connections of the computer which has been installed on in addition to act as IDS. The IDS records all incoming and outgoing packages in tcpdump binary format. The use of this standard format simplifies all data manipulations.

## **The Detection process**

By initializing the system, the IDS detect the attack and transfer the information to honey pot in the main network, and then Honey Pot is ready to interact with the suspected adversaries. When the IDS in external agents detect an attack, they report the event to the honey pot which will decide how to manage it. Such attacks can be listed as follows; Nimda, Spoofing. Attempts to execute cmd.exe, Attempts to open a new network socket in order to download further hacking utilities, A worm that starts on each infected system an email relying server which it uses for its further spreading like Nimda and Denial of Services. Honey pot orders switch to send a copy of all packages which their destination is the said agent, and also commands the agent not to answer, thus will receive the trail of suspended threat packages. All of the process must be hidden to the end-user. Indeed, the connection will be shifted to the Honey Pot. Before processing, the dispatcher of the Honey Pot queries the configuration database to find a previous configuration that corresponds to the destination IP address. If there is no specific configuration, a default template is used. Thus a packet together with corresponding configuration is handed to the verified protocol. The ICMP protocol supports most ICMP messages. By default, all honey pot configurations are able to respond to echo requests and process unreachable destination



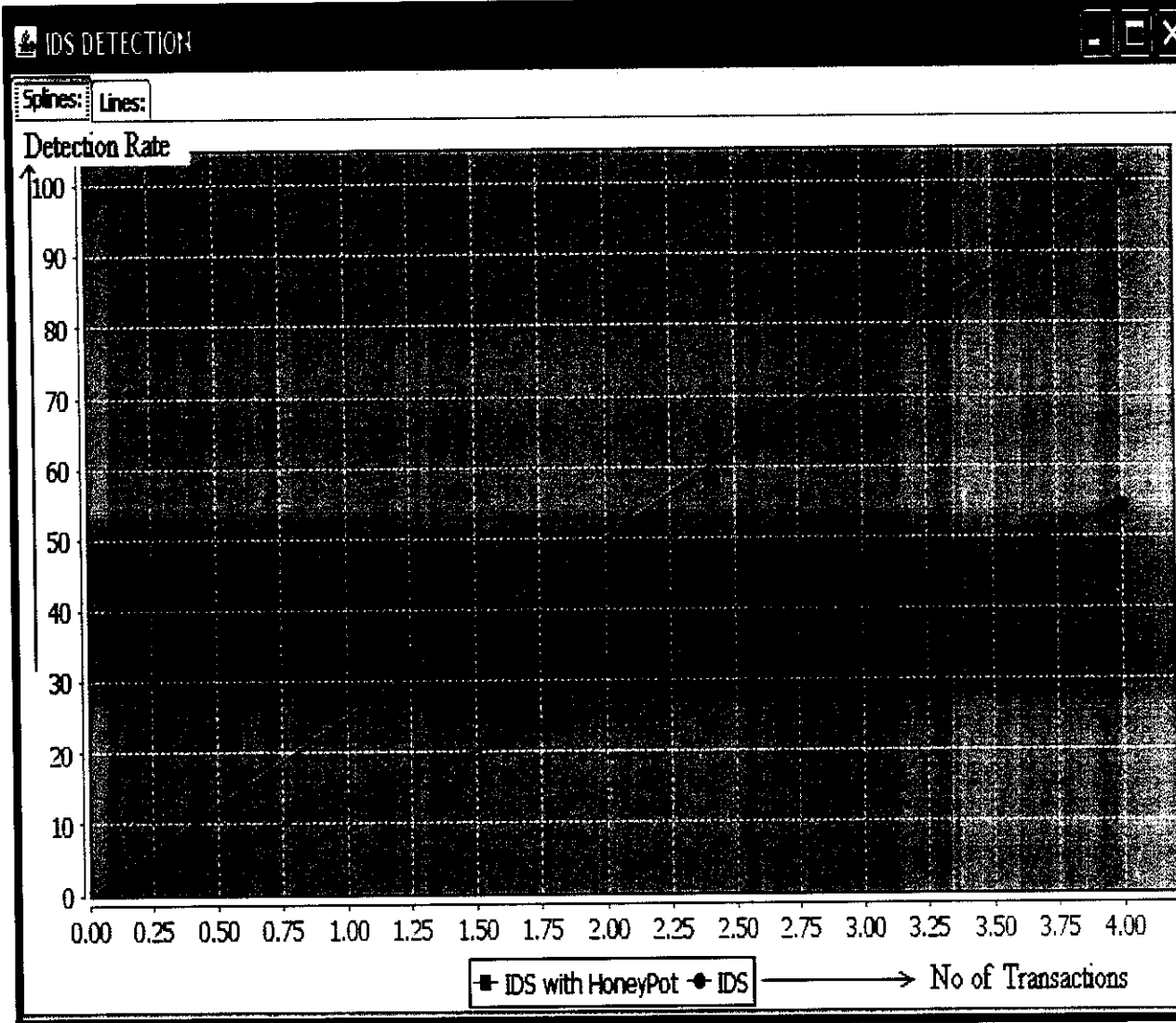
messages. For TCP and UDP, the framework establishes a connection to arbitrary services.

When a connection request is received, the protocol checks if the packet is established before, then any new data is sent to the already initiated service application. If the packet request is new, a new process is created to run the appropriate service application. The corresponding protocol responds like a normal network to the end-user. If the Honey Pot found out that the detection was wrong, without breaking up the connection via IDS, it will ask external agents to carry on connection. It will also send all saved information relevant to the connection to the previous destination. Therefore the number of false positives will be decreased. It is a crucial factor to show the precision and accuracy of the detection.



## 5. PERFORMANCE EVALUATION

### Comparison of False Alarm Rate: IDS with Honey Pot and IDS





## 6. CONCLUSION

This project presents a new system on a distributed agent based network. The system, in case of suspended adversaries, changes the path of connection to the honey pot system for a more close investigation. Decreasing false alarms rate is the system's objective. The system also tries to store any necessary information about the identified attacker. In the simulated system we obtained good results which were declaring the decreasing in false positive ratio. Due to the filtering of IDS the load of the system directed to the Honey Pot is decreased leading the Honey Pot to better performance in packet dropping. We tested the proposed system with different number of packets we generated.

In order to test the scalability of the system we did the testing with three different network loads which were the rate for generation of the packets. We should pay more attention to the following problems: How to shift the connection with the all related data in more details. We have to expose director to more aggressive traffic patterns to get a better understanding of its performance. The system has intelligent to deal with the data communication between honey pot and the end user.



## 7. FUTURE WORK

Tests show that the proposed system can manage to improve the detection accuracy in Comparison with the conventional honeyd and IDS. As it is clear, in the proposed system the objective is to recheck the attacking packets in order to avoid the false positives while both honeyd and IDS have their own false positive ratio which is definitely greater than what we gained as a cooperation of these two systems.

In future the attacks are dynamically updated the all the attacks such as Massive scans, Targeted attacks, IRC wars, Unusual root kits, Covert back doors ,New exploits, Worm spreading The Hybrid design could even detect attack against IDS control centers while agents roam through the network to spot intrusions. Nonetheless, weaknesses are unavoidable in a new design, and many areas discussed in this paper would benefit from further efforts to clarify the design fine points and implementation details. Further work should also look into mobile agent's intercommunication and negotiation which can help investigative mobile agents to share their knowledge. In addition intrusion pattern's knowledge sharing between IDS control centers can be considered for further studies.





## 8. APPENDIX

### 8.1 SAMPLE SOURCE CODE

#### SOURCEGUIMAIN.java

```
import java.awt.Dimension;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.beans.PropertyVetoException;

import javax.swing.*;

class SourceGUIMain extends JFrame implements ActionListener

{

    private static final long serialVersionUID = 1L;

    JMenuBar menuBar = new JMenuBar();

    JMenu mnuFile,mnuTools,mnuProcess;

    JMenuItem mtmExit,mtmRouting;

    JMenuItem fileTransfer,pktTrans;

    static JDesktopPane deskView;

    public SourceGUIMain() {

        toInitialize();

        setDefaultCloseOperation(EXIT_ON_CLOSE);
```

```

        Toolkit.getDefaultToolkit().getScreenSize();

setSize(dimension.width,dimension.height-30);

setTitle("IMPROVED INTRUSION DETECTION SYSTEM");

setJMenuBar(addMenuBar());

setContentPane(deskView);

setVisible(true);
}

public static void main(String[] args) {

    new SourceGUIMain();

}

private JMenuBar addMenuBar() {

    mnuFile = new JMenu("File");

    mtmExit = new JMenuItem("Exit",'x');

    mnuFile.add(mtmExit);

    mnuFile.setMnemonic('F');

    menuBar.add(mnuFile);

    mnuProcess = new JMenu("Process");

    fileTransfer = new JMenuItem("File Transfer",'T');

    pktTrans = new JMenuItem("PacketTransmission",'P');

    mnuProcess.add(fileTransfer);

    mnuProcess.add(mnuProcess);
}

```

```
mnuProcess.add(pktTrans);

menuBar.add(mnuProcess);

mnuTools = new JMenu("Tools");

mtmRouting = new JMenuItem("Route Config",'r');

mnuTools.add(mtmRouting);

mnuTools.setMnemonic('T');

menuBar.add(mnuTools);

mtmExit.addActionListener(this);

mtmRouting.addActionListener(this);

pktTrans.addActionListener(this);

fileTransfer.addActionListener(this);

return menuBar;
}

private void toInitialize() {

    try {

        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

    } catch (Exception e) {

        e.printStackTrace();

    }

    deskView = new JDesktopPane();
}
```

```

public void actionPerformed(ActionEvent e) {
    if(e.getSource()==pktTrans){
        PacketTransmission pt=new PacketTransmission();
        deskView.add(pt);
        try {
            pt.setSelected(true);
        } catch (PropertyVetoException e1) {
            e1.printStackTrace();
        }
    }
    if(e.getSource()==fileTransfer){
        FileTransmission ft=new FileTransmission();
        deskView.add(ft);
        try {
            ft.setSelected(true);
        } catch (PropertyVetoException e1) {
            e1.printStackTrace();}
    }
}
}

```

## **INTRUSIONDETECTION.java**

```
import java.io.DataInputStream;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Date;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
public class IntrusionDetection implements Runnable {
    Connection con;
    Statement st;
    ResultSet rs;
    String getRecIp = "";
```

```

ServerSocket routerServer;

Socket ss;

DataInputStream dis;

DataOutputStream dos;

Thread readThread;

String readingString="";

String fname="";

JFrame par=null;

public IntrusionDetection(JFrame parent){

    par=parent;

    IDSStatusPanel.txtInforamtion.append("\n Intrusion Detection ");

    IDSStatusPanel.txtInforamtion.append("\n Started..... ");

    System.out.println("Router Started at the port ");

    try {

        routerServer=new ServerSocket(2001);

while(true){

routerSocket=routerServer.accept();

System.out.println("Router is Connected...");

dis=new DataInputStream(routerSocket.getInputStream());

dos=new DataOutputStream(routerSocket.getOutputStream());

        readThread=new Thread(this);

        readThread.start();

```

```

} catch (IOException e) {
e.printStackTrace();
}}

public static void main(String[] args) {
    //new IntrusionDetection();}

public void run() {
try {
while(dis.available()>0){
readingString=dis.readUTF();
System.out.println("Reading Data:"+readingString);
String mode=readingString.substring(readingString.lastIndexOf("#")+1);
System.out.println("The Operate Mode:"+mode);
if(mode.equals("Packet")){
System.out.println("Packet Transmission Message :"+readingString);
String rMsg=readingString.substring(0,readingString.indexOf("#"));
String
rIp=readingString.substring(rMsg.length()+1,readingString.indexOf("@"));
String
rPort=readingString.substring(rMsg.length()+rIp.length()+1,readingString.lastIndexOf("#"));
IDSStatusPanel.txtInforamtion.append("\n Data Received:"+rMsg);
getRecIp =""+routerSocket.getRemoteSocketAddress();
IDSStatusPanel.txtInforamtion.append("\n SenderInfo (IP,PORT) :"+getRecIp);
IDSStatusPanel.txtInforamtion.append("\n Packet Length:"+rMsg.length());

```

```

IDSStatusPanel.txtInforamtion.append("\n Receiver Info (IP,PORT)
:"+getMsg[1].split("@")[0]+","+ getMsg[1].split("@")[1]);

String getConfig=readingString.split("@")[1];

startForwarding(getMsg[0],getMsg[1],mode);

}

else{

System.out.println("The Message is paya:"+readingString);

String rMsg=readingString.substring(0,readingString.indexOf("#"));

String
rIp=readingString.substring(rMsg.length()+1,readingString.indexOf("@"));

String getMsg[]=readingString.split("#");

fname=rMsg;

System.out.println("File Name :"+ fname);

String faddrss=dis.readUTF();

String remip="" +routerSocket.getInetAddress();

IDSStatusPanel.txtInforamtion.append("\nTheReceiving
Address:"+routerSocket.getInetAddress());

IDSStatusPanel.txtInforamtion.append("\nTheReceiving
Port:"+routerSocket.getPort());

System.out.println("Receiving Address :"+ faddrss);

InputStream is = routerSocket.getInputStream();

System.out.println("The Message:"+getMsg[0]);

System.out.println("The address:"+getMsg[1]);

System.out.println("The Mode:"+mode);

```



```

checkIntrusions(""+routerSocket.getInetAddress(),mode);
FileOutputStream inFile = new FileOutputStream(fname);

int ch = 0;

while ((ch = is.read()) != 0) {

inFile.write(ch);}

inFile.close();}

Thread.sleep(1000);

}

} catch (Exception e) {

e.printStackTrace();}

}

public void startForwarding(String readData,String config,String mode){

    System.out.println("Data is Forwarding...");

    String chkPro[]=mode.split(",");

    System.out.println("Chk Pro :"+chkPro[0]);

    if(chkPro[0].equals("FileTransfer")){

        try{

            String getMsg[]=readingString.split("#");

            String ip=config.split("@")[0];

            int port=Integer.parseInt(config.split("@")[1]);

            System.out.println("The ip is:"+ip+" Port:"+port);

            IDSStatusPanel.txtInforamtion.append("\n Origianl Packet:"+readData);

```

```

IDSStatusPanel.txtInforamtion.append("\nReceiverInfo(IP,PORT):"+getMsg[1].split("@")[0]+", "+ getMsg[1].split("@")[1]);

boolean result=checkIntrusions(mode);

if(result==true){

JOptionPane.showMessageDialog(par,"some Intrusion Found..\nThis attacks are tranfered into Honeypot ");

String hip=getHoneyIp();

ss=new Socket(hip,7001);

dos=new DataOutputStream(ss.getOutputStream());

IDSStatusPanel.txtInforamtion.append("\n Packet is Honeyed:");

System.out.println("In Intrusions");

System.out.println("The Data:"+readData);

System.out.println("The Config:"+config);

System.out.println("The Mode:"+mode);

System.out.println("The Hac IP:"+getRecIp.substring(1,getRecIp.indexOf(":")));

dos.writeUTF(readData+"#" +config+"#" +mode+"*" + "192.168.1.31");

}

else {ss=new Socket(ip,port);

dos=new DataOutputStream(ss.getOutputStream());

dos.writeUTF(readData+", "+mode);

}}catch(Exception ex){

ex.printStackTrace();

}

```

```

}

if(chkPro[0].equals("Packet")){

    try {

        String ip=config.split("@")[0];

        int port=Integer.parseInt(config.split("@")[1]);

        //ip Spoofing Checking...

        System.out.println("Received Ip:"+ip);

        boolean result=scanAddress(ip);

        IDSStatusPanel.txtInforamtion.append("\n    The    Ip    Scanning
Result:"+result);

        if(result==true) {

IDSStatusPanel.txtInforamtion.append("\n The Spoofed IP :"+ip);

JOptionPane.showMessageDialog(par,"ip Spoofing Attack Found...");

JOptionPane.showMessageDialog(par,"some Intrusion Found...\nThis attacks are
tranfered into Honeypot ");

ss=new Socket(getHoneyIp(),7001);

System.out.println("Remote Address " + ss.getRemoteSocketAddress());

dos=new DataOutputStream(ss.getOutputStream());

IDSStatusPanel.txtInforamtion.append("\n Packet is Honeyed:");

dos.writeUTF(readData+"#" +config+"#" +mode+"*" +getRecIp.substring(1,getRe
cIp.indexOf(":")));

}

Else{

ss=new Socket(ip,port);

```

```

dos=new DataOutputStream(ss.getOutputStream());
IDSStatusPanel.txtInforamtion.append("\n Origianl Packet:"+readData);
                                dos.writeUTF(readData+","+mode);
                                }
                                }
                                catch (UnknownHostException e) {
                                    e.printStackTrace();
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                                }}}

```

```

public void fileTransfer(String file) {
try {
                                OutputStream os = ss.getOutputStream();
                                File f = new File(file);
                                System.out.println("File Name is:"+f);
                                FileInputStream in = new FileInputStream(f);

                                int i = 0;
                                System.out.println("\nSending File ...");
                                while ((i = in.read()) != -1) {
                                    os.write(i);
                                    os.flush();
                                }
}
}

```

```

        in.close();

    } catch (Exception e) {

        System.out.println("\nFileException" + e);

    }}

catch(Exception e){

e.printStackTrace()}}

public boolean scanAddress(String getIp){

    boolean spoof=false;

    int count=0;

    setDbConnection();

    try{

        String str="";

        st=con.createStatement();

        String qry="select * from HCF";

        rs=st.executeQuery(qry);

        while(rs.next()){

str=rs.getString(1)+","+rs.getString(2)+","+rs.getString(3);}

        String ipList[]=str.split(",");

        String str23=getRecIp.substring(1,getRecIp.indexOf(":"));

        System.out.println("getIP:"+str23);

        catch(Exception ex){

```

```

        }if(count>0){
            spoof=true;}
    else{
        spoof=false;}
    return spoof;}

    public boolean checkIntrusions(String mode){
        boolean res=false;

        try {
            String datamode[]=mode.split(",");
            if(datamode.length>1){
                res=checkDosAttack(datamode[1]);
            }
        } catch (Exception e) {e.printStackTrace();}

    return res;}

    public boolean checkDosAttack(String attc){
        boolean res=false;

        if(attc.equalsIgnoreCase("dos")){
            JOptionPane.showMessageDialog(par,"DOS Attack Has been Found...");
            res=true;
        }

        if(attc.equalsIgnoreCase("nim")){
            JOptionPane.showMessageDialog(par,"Nimda Attack Has been Found...");
            res=true;
        }
    }
}

```

```

        }if(attc.equalsIgnoreCase("exe")){
JOptionPane.showMessageDialog(par,"Exe Attack Has been Found...");
        res=true;}
return res;
new SimpleRead();
}
public String getIdsIp(){
    String str1="";
    try{
        setDbConnection();
        st=con.createStatement();
        String query="select * from IpSettings";
        rs=st.executeQuery(query);
        while(rs.next()){
            str1=rs.getString("IDSip");} }
    catch(Exception e){
        e.printStackTrace();}return str1;}

```

### **HoneyPot.java**

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

```

```
import java.net.Socket;
import java.net.UnknownHostException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class HoneyPot implements Runnable {
    ServerSocket honeySocket;
    Socket ss;
    DataInputStream dis;
    DataOutputStream dos;
    File f=new File("Script.hon");
    Thread t;
    Connection con;
    Statement st;
    ResultSet rs;
    JFrame par;
    int findAttacks=0;
    int allowAttacks=0;
    int blockAttack=0;
    public HoneyPot(JFrame frame) {
        par=frame;
```



```

honeySocket=new ServerSocket(7001);
HoneyStatusPanel.txtInforamtion.append("\nHoney Pot Program is Started...");

    while(true){

        ss=honeySocket.accept();

        dis=new DataInputStream(ss.getInputStream());

        System.out.println("Varifing Script Files....");

        t=new Thread(this);

        t.start();

    }

    catch (IOException e) {

        e.printStackTrace();}}

public void run() {

    try {

        String readingString=dis.readUTF();

        System.out.println("File Data:"+readingString);

        JOptionPane.showMessageDialog(par,"HoneypotScanningthe Attack...");

        String chkIp=readingString.substring(readingString.lastIndexOf("*")+1);

        String rMsg=readingString.substring(0,readingString.indexOf("#"));

String
rIp=readingString.substring(rMsg.length()+1,readingString.indexOf("@"));

String
rPort=readingString.substring(readingString.indexOf("@")+1,readingString.inde
xOf("@")+5);

String

```

```
HoneyStatusPanel.txtInforamtion.append("\n The Message:"+rMsg);
HoneyStatusPanel.txtInforamtion.append("\n The IP:"+rIp);
HoneyStatusPanel.txtInforamtion.append("\n The Port:"+rPort);
HoneyStatusPanel.txtInforamtion.append("\n The Checking IP:"+chkIp);
java.util.Date dte=new java.util.Date();
SimpleDateFormat dformat=new SimpleDateFormat("dd/MM/yyyy :");
setDbConnection();
try{
st=con.createStatement();
String query="";
System.out.println(Type.length);
if(Type.length>1){
query="insert into AttackDetails values('"+ dte +"','"+ rMsg +"','"+ rIp +"','"+
rPort +"','"+ Type[0] +"','"+ Type[1] +"')";}
else{query="insert into AttackDetails values('"+ dte +"','"+ rMsg +"','"+ chkIp
+"','"+ rPort +"','"+ Type[0] +"','Spoon')";
}
st.executeUpdate(query);
findAttacks++;
boolean bool=checkScript(chkIp);
if(bool==true){
allowAttacks++;
readingString=readingString.substring(0,readingString.lastIndexOf("*"));
```



```

        e.printStackTrace();
    }
    public void setDbConnection() {
        try {
            String driver="sun.jdbc.odbc.JdbcOdbcDriver";
            String path="spoofing.mdb";
            Stringconnection="Jdbc:Odbc:Driver={MicrosoftAccessDriver
(*.mdb)};DBQ="+path;Class.forName(driver);
            con=DriverManager.getConnection(connection);
        }
        catch(Exception e){
            e.printStackTrace();}
    }public boolean checkScript(String ipString) {
        boolean res=false;
        setDbConnection();
        try {
            st=con.createStatement();
            String query="select * from ScriptDetails where ip='"+ ipString +"'";
            rs=st.executeQuery(query);
            while(rs.next()) {
                String str=rs.getString(4);
                if(str.equals("YES")){
                    res=true;}
            }
        }
    }

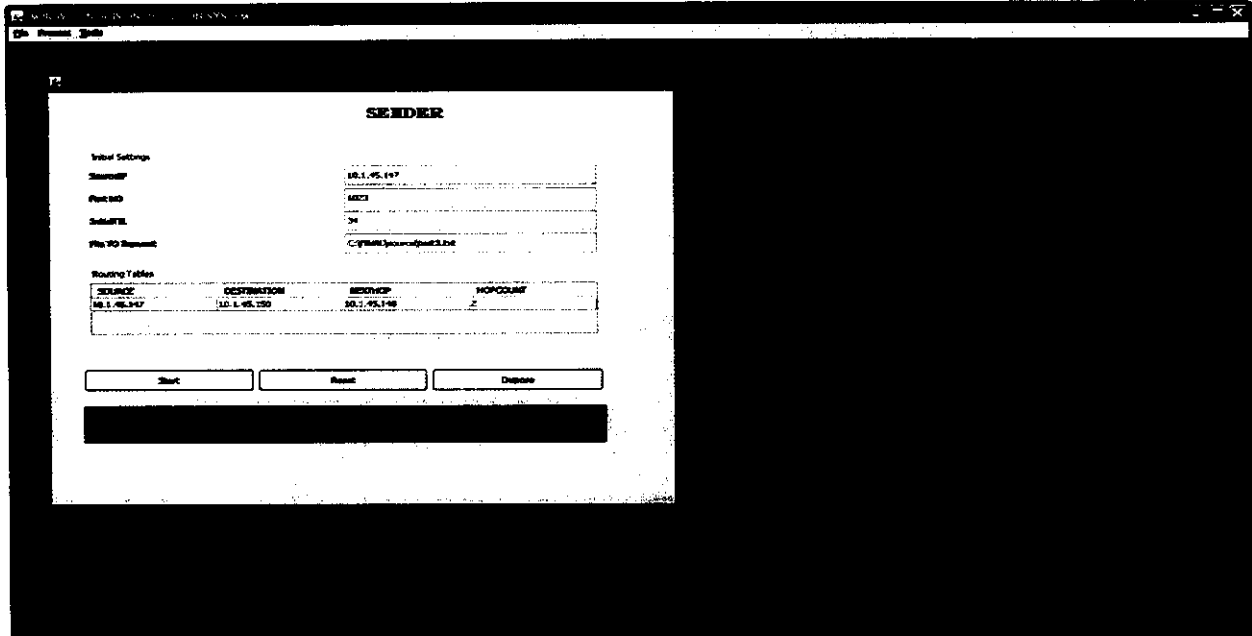
```

```
        res=false;}}}  
        catch(Exception ex){  
            ex.printStackTrace();  
        }  
        return res;  
    }  
    public String getNextIp(){  
        String str1="";  
        try{  
            setDbConnection();  
            st=con.createStatement();  
            String query="select * from HCF";  
            rs=st.executeQuery(query);  
            while(rs.next()){  
                str1=rs.getString(2);  
            }  
        }  
        catch(Exception e){  
            e.printStackTrace();  
        }  
        return str1;}}
```

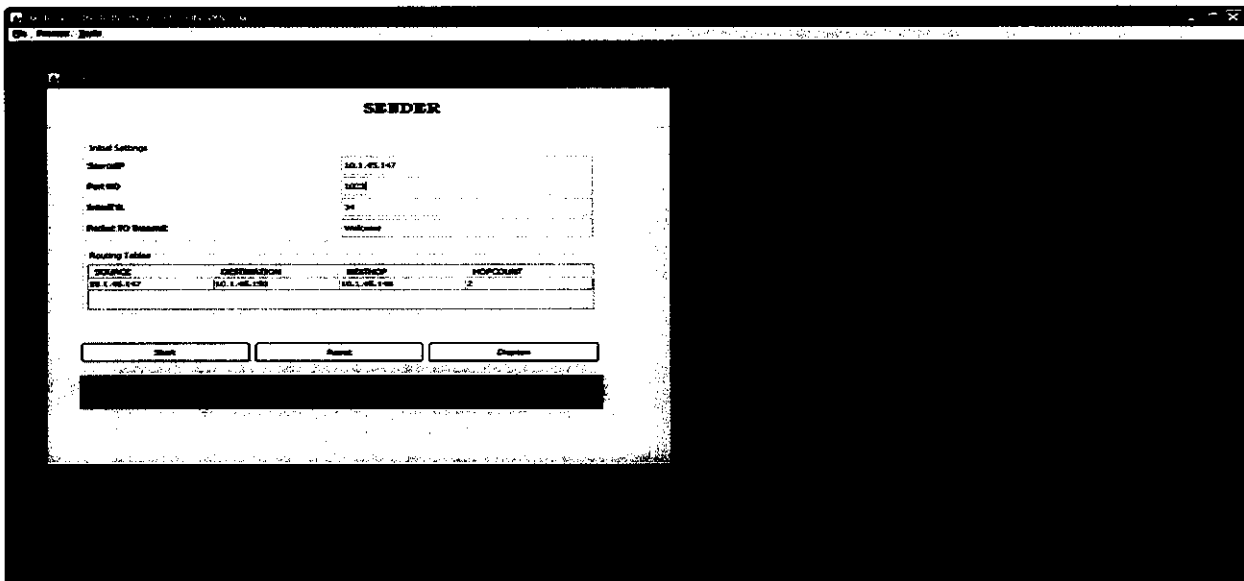
## 8.2 SCREEN SHOTS

### SENDER

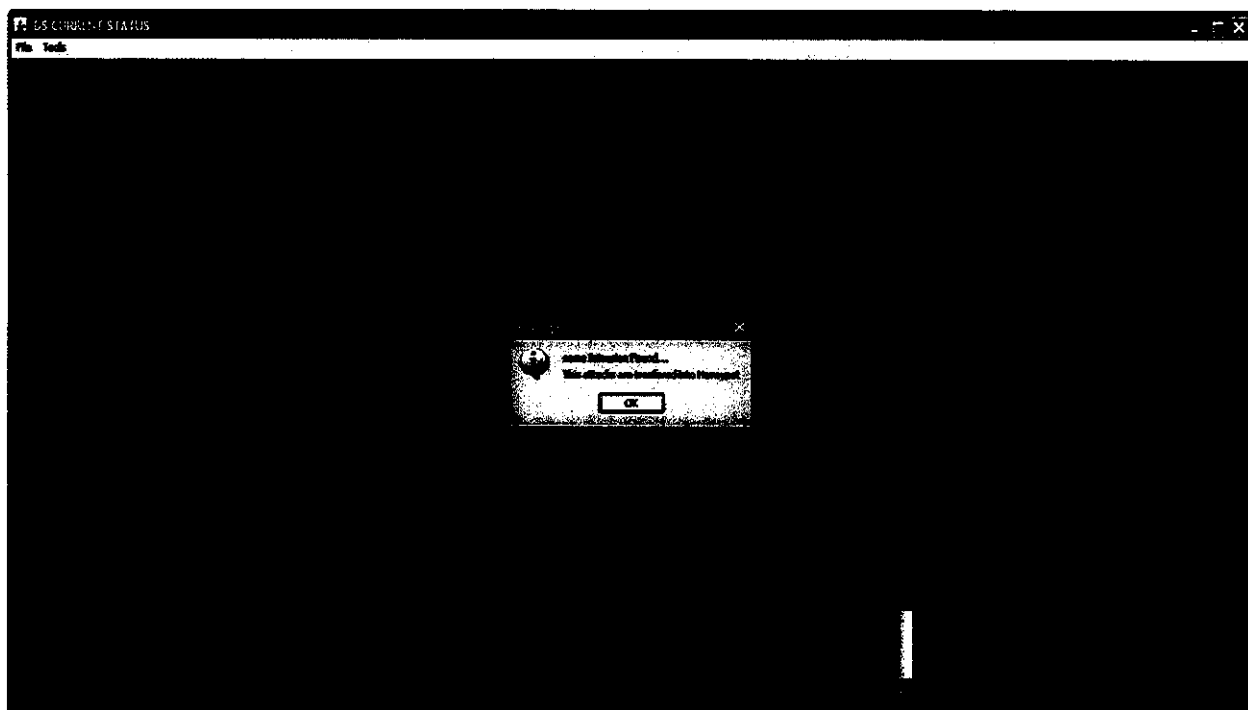
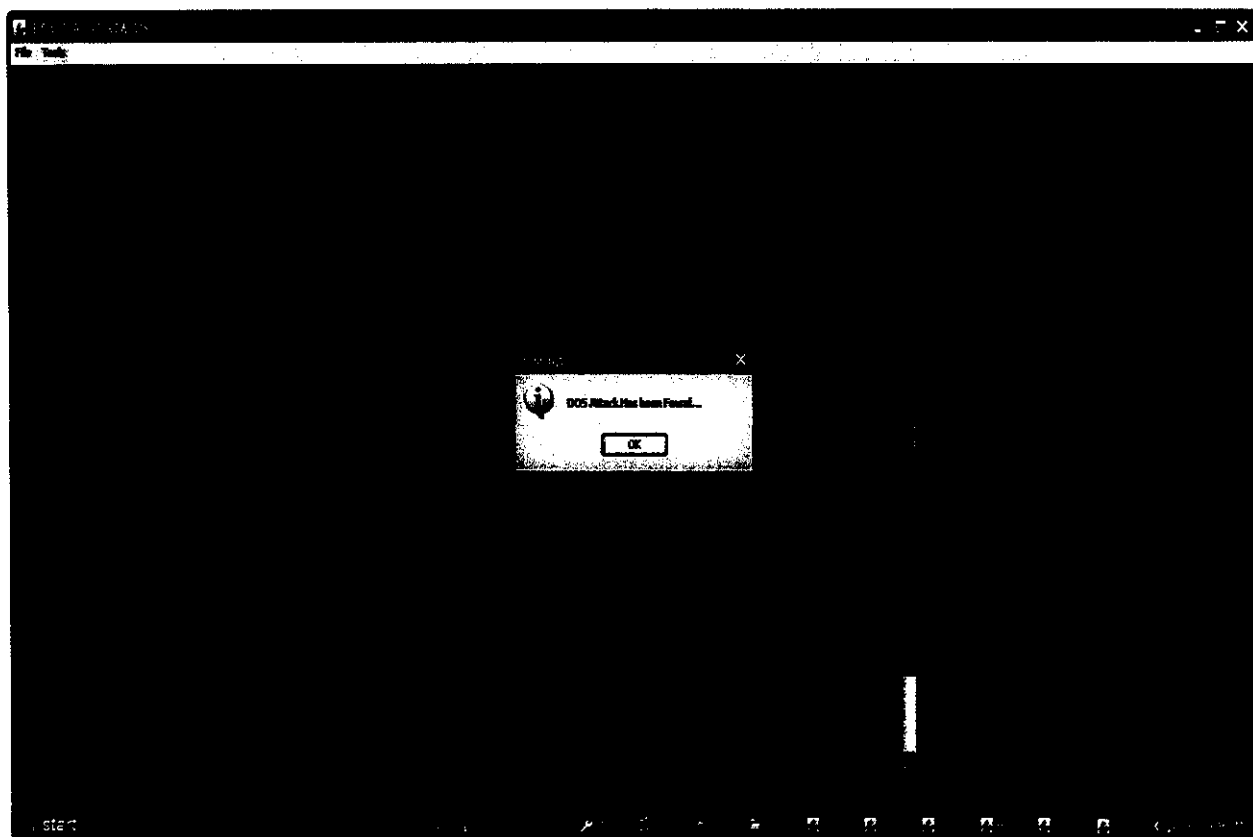
#### 1. File Transmission



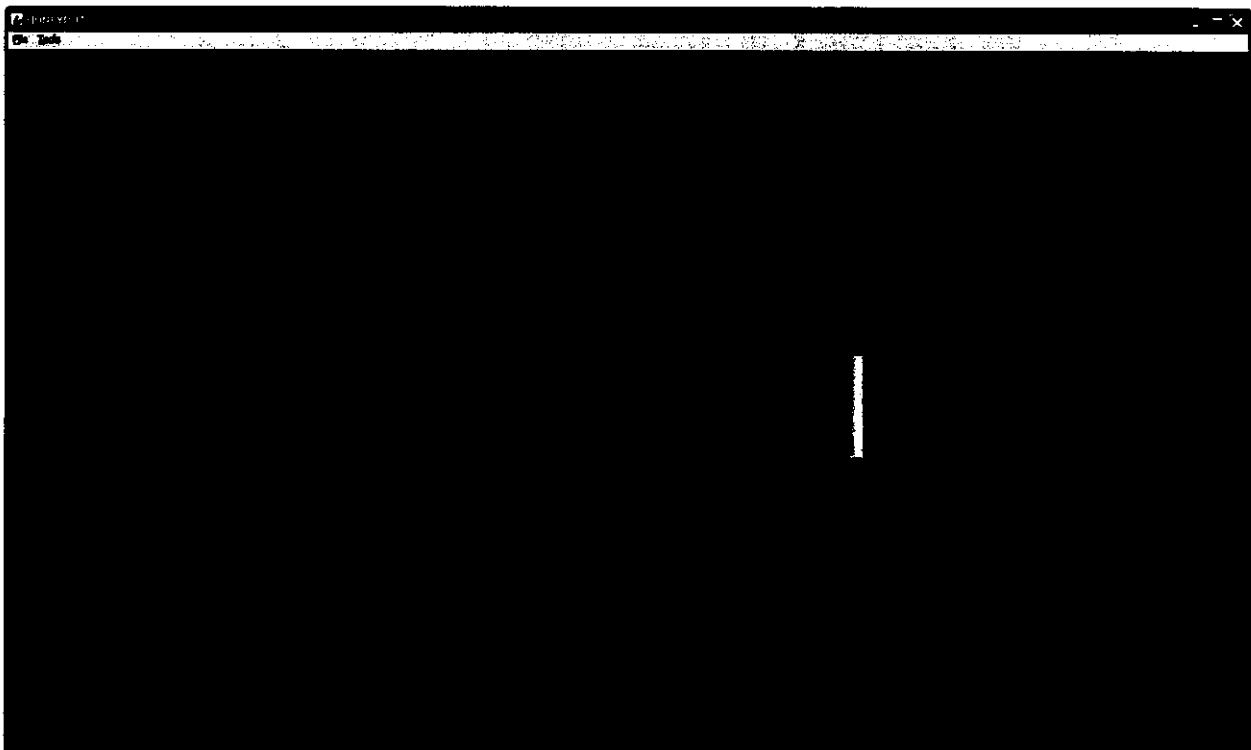
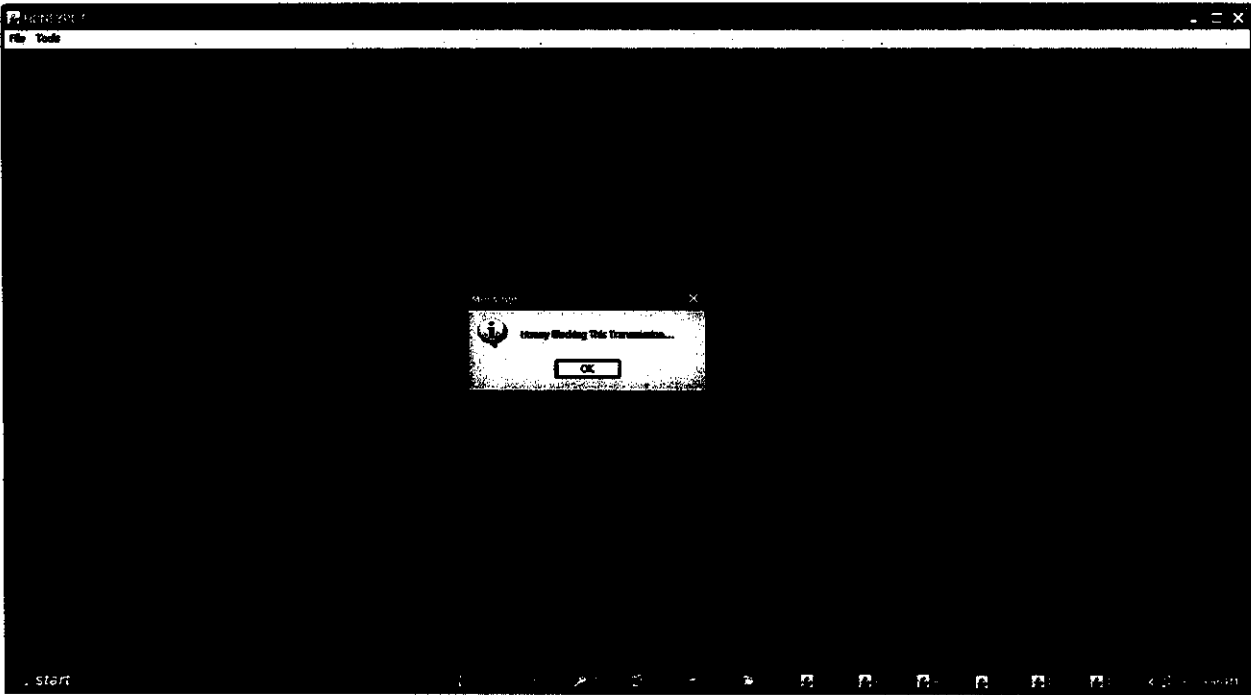
#### 2. Packet Transmission



# INTRUSION DETECTION SYSTEM (IDS)

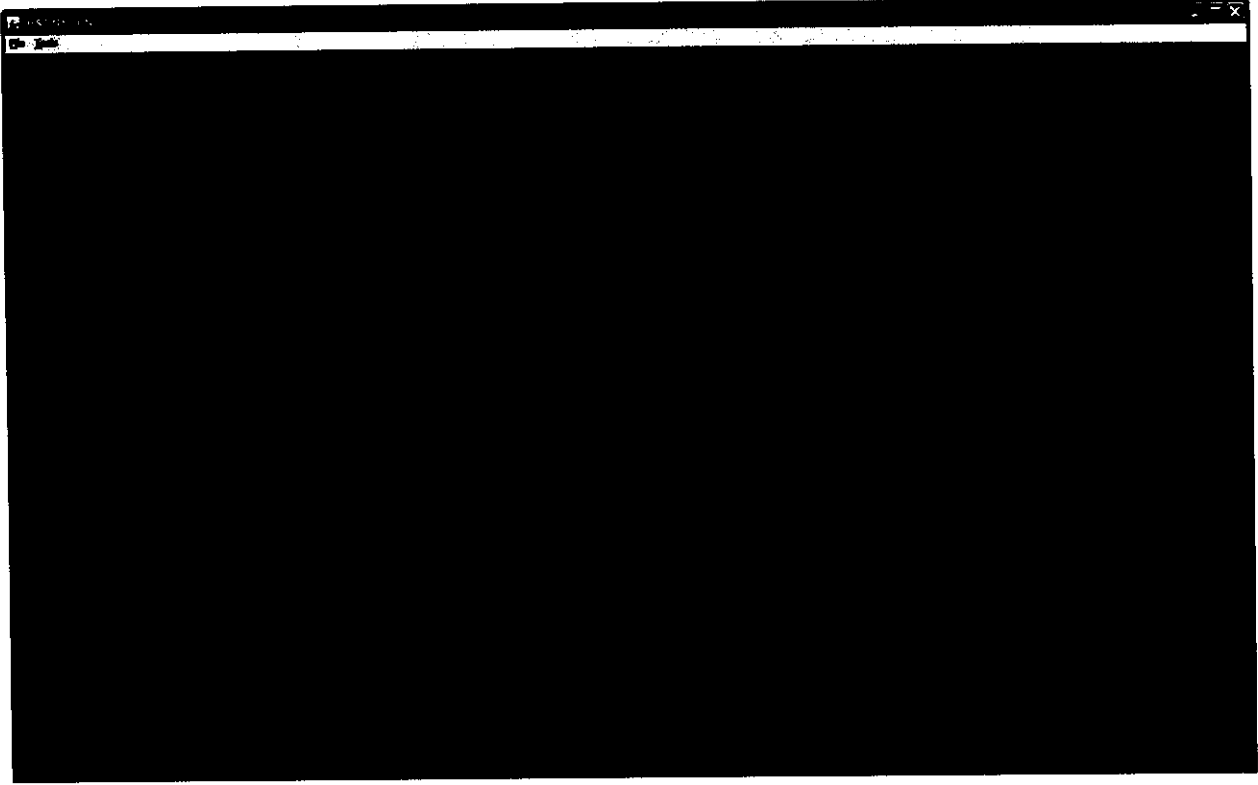


# HONEY POT

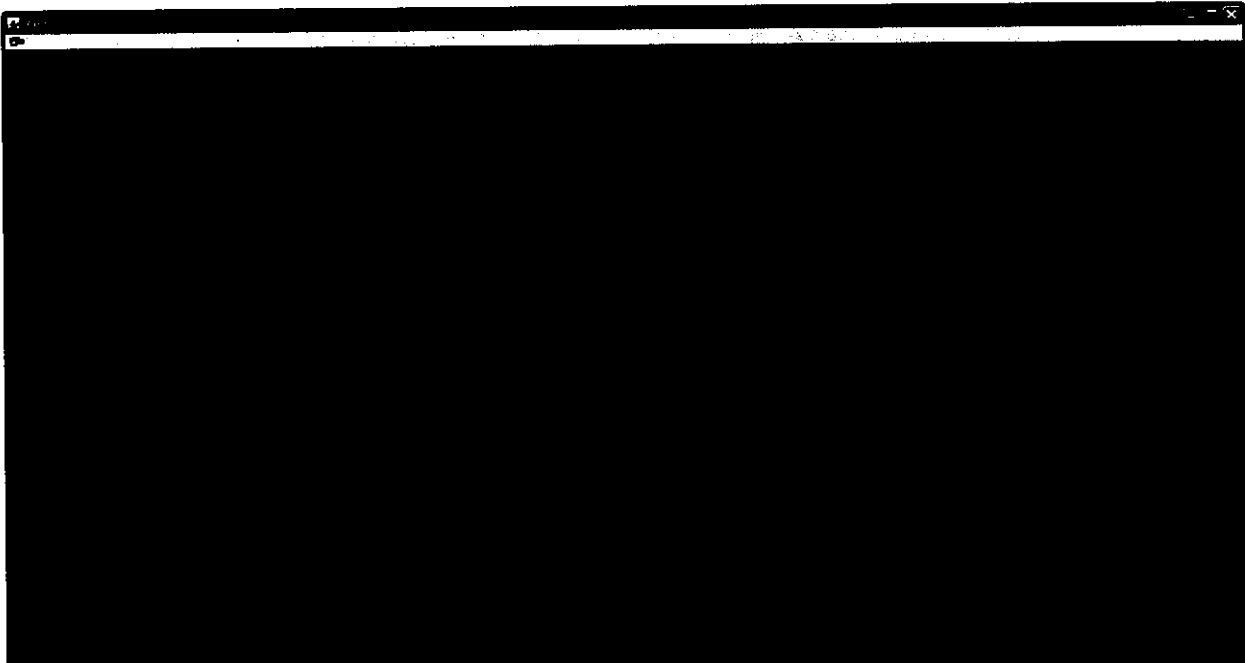




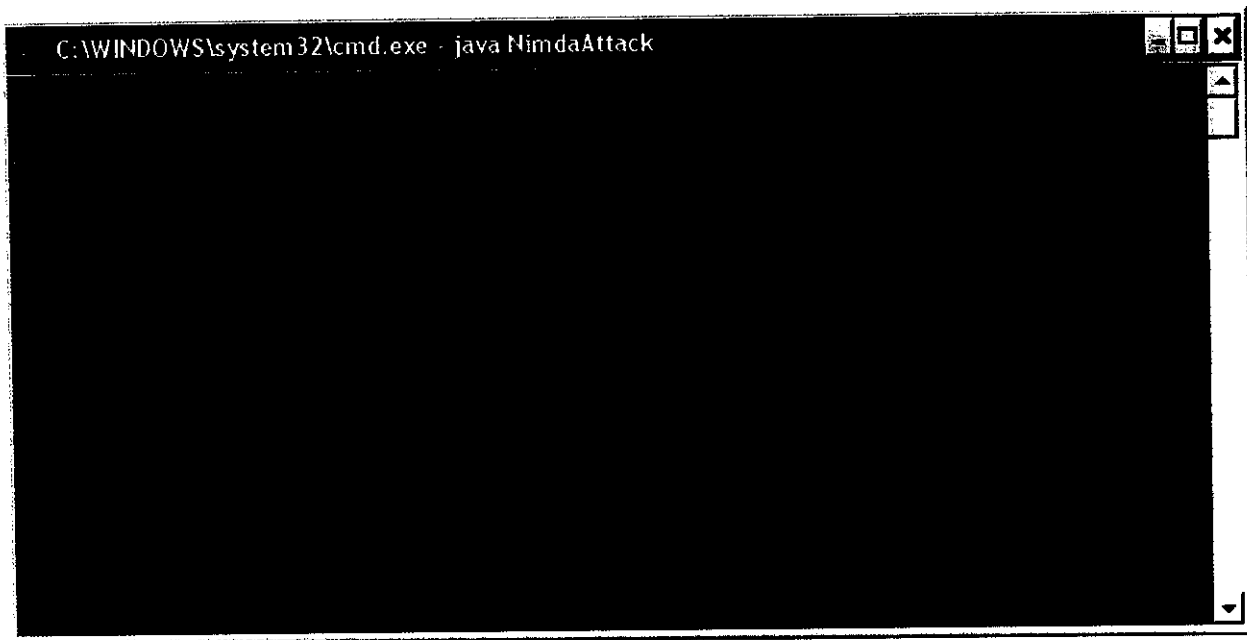
## DESTINATION



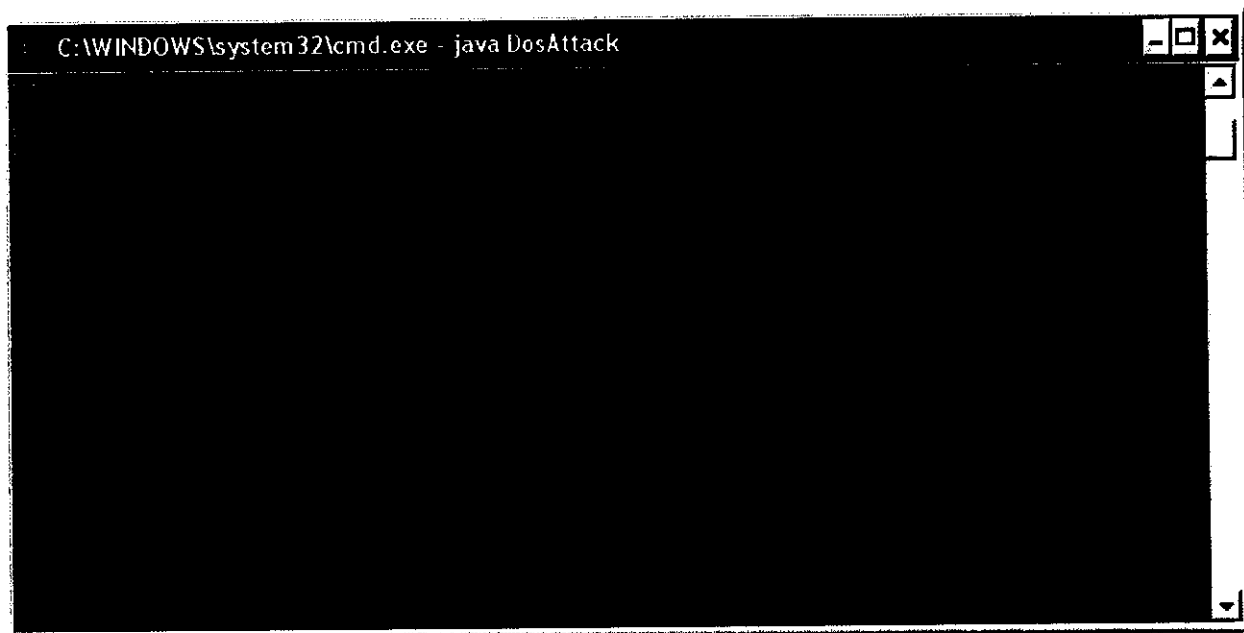
## ROUTER



## NIMDA ATTACK



## DENIAL OF SERVICE (DOS) ATTACK



# RECORDS

## LOG FILES

Table Tools    AttackDetails - Microsoft Access

Home    Tables

View    Paste    Callion    Refresh All    X Delete    More    Selection    Filter    Size to Fit Form    Switch Windows    Find

HDate	Hmsg	Hip	Thread	Infection
Sat Jan 16 05:41:28	E:\JavaPrograms\Hone	192.168.1.21 1031	FileTransfer	Dos
Sat Jan 16 06:41:29	E:\JavaPrograms\Hone	192.168.1.21 1031	FileTransfer	Dos
Sat Apr 03 19:27:35	macha hacking	192.168.1.21 1031	Packet	Spoof
Sat Apr 03 19:29:14	Dai sent	192.168.1.21 1031	Packet	Spoof
Sat Apr 03 19:33:21	hai	192.168.1.21 1031	Packet	Spoof
Sat Apr 03 19:33:55	doubt	192.168.1.21 1031	Packet	Spoof
Sat Apr 03 21:41:11	sdfsdfsd	192.168.1.31 1031	Packet*192.168	Spoof
Sat Apr 03 21:41:37	sdfsdfsd	192.168.1.31 1031	Packet*192.168	Spoof
Sat Apr 03 21:41:46	sdfsdfsd	192.168.1.31 1031	Packet*192.168	Spoof
Sat Apr 03 21:42:26	sdfsdfsd	192.168.1.31 1031	Packet*192.168	Spoof
Sat Apr 03 21:42:46	sdfsdfsd	192.168.1.31 1031	Packet*192.168	Spoof
Sat Apr 03 21:53:41	E:\JavaPrograms\Hone	192.168.1.21 1031	FileTransfer	Dos
Sat Apr 03 22:33:31	E:\JavaPrograms\Hone	192.168.1.21 1031	FileTransfer	nim
Sat Apr 03 22:39:55	welcome	192.168.1.31 1031	Packet*192.168	Spoof
Sat Apr 03 22:39:55	welcome	192.168.1.21 1031	Packet*192.168	Spoof
Sat Apr 03 22:41:06	E:\JavaPrograms\Hone	192.168.1.21 1031	FileTransfer	nim
Sat Apr 03 23:00:25	E:\JavaPrograms\Hone	192.168.1.21 1031	FileTransfer	nim
Sat Apr 03 23:02:45	E:\JavaPrograms\Hone	192.168.1.21 1031	FileTransfer	nim
Sun Apr 04 10:49:1	test2.txt	192.168.1.21 1031	FileTransfer	nim*192.168.1
Sun Apr 04 10:52:1	test3.txt	192.168.1.21 1031	FileTransfer	nim*192.168.1
Sun Apr 04 10:55:4	test2.txt	192.168.1.21 1031	FileTransfer	nim*192.168.1
Sun Apr 04 10:56:2	test2.txt	192.168.1.21 1031	FileTransfer	nim*192.168.1
Sun Apr 04 10:58:3	test2.txt	192.168.1.21 1031	FileTransfer	nim*192.168.1
Sun Apr 04 10:59:1	test3.txt	192.168.1.21 1031	FileTransfer	nim*192.168.1
Sun Apr 04 16:33:2	test2.txt	192.168.1.21 1031	FileTransfer	exe*192.168.1
Sun Apr 04 16:33:3	test2.txt	192.168.1.21 1031	FileTransfer	exe*192.168.1

Record: 11 of 53    Search

start    10:58 AM



## 9.REFERENCES

- [1] Babak Khosravifar, Jamal Bentahar ,” An Experience Improving Intrusion Detection Systems False Alarm Ratio by Using Honey pot” , IEEE 22nd International Conference on Advanced Information Networking and Applications,2008.
- [2] Yun Wang, Xiaodong Wang, Bin Xie, Demin Wang and Dharma P. Agrawal, “Intrusion Detection in Homogeneous and Heterogeneous Wireless Sensor Networks”, IEEE transactions on mobile computing, June 2008.
- [3] Karthik Sadasivam, Banuprasad Samudrala, T. Andrew Yang,” Design Of Network Security Projects Using Honeypots”, CCSC: South Central Conference, 2005 Consortium for Computing Sciences in Colleges.
- [4] Babak Khosravifar, Maziar Gomrokchi, Jamal Bentahar ,” A Multi-Agent-based Approach to Improve Intrusion Detection Systems False Alarm Ratio by Using Honeypot”, IEEE International Conference on Advanced Information Networking and Applications Workshops, 2009.
- [5] Wang Zhenqi, Wang Xinyu,” NetFlow Based Intrusion Detection System”, IEEE International Conference on MultiMedia and Information Technology , 2008.
- [6] Liberios Vokorokos,Alzbeta Kleinova, Ondrej Latka,” Network Security on the Intrusion Detection System Level”, IEEE Conference on Networks, 2006.
- [7] Bin Dong, Xiu-Ling Liu,” An Improved Intrusion Detection System Based On Agent”,IEEE Proceedings of the Sixth International Conference on Machine