P-3107

A NEW DATA-MINING BASED APPROACH FOR NETWORK INTRUSION DETECTION

A PROJECT REPORT

Submitted by

EALUMALAI G 71206104013

GOKULAPRABUKUMAR G N 71206104015

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2010

BONAFIDE CERTIFICATE

Certified that this project report **"A NEW DATAMINING BASED APPROACH FOR NETWORK INTRUSION DETECTION"** is the bonafide work of **"EALUMALAI G"** and **"GOKULAPRABUKUMAR G N "** who carried out the project under my supervision.

[illegible]

SIGNATURE

Dr. S. Thangasamy

DEAN

Department of Computer Science & Engg
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore – 641606

[illegible]

SIGNATURE

Ms. C. Ramathilagam

LECTURER

Department of Computer Science & Engg
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore – 641606

The candidates with University Register Nos. **71206104013** and **71206104015** were examined by us in the project viva-voce examination held on ......16.4.2010..............

INTERNAL EXAMINER

[illegible]

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made this possible and whose constant guidance and encouragement crowns all efforts with success.

We are extremely grateful to **Dr. S. Ramachandran**, Principal, Kumaraguru College of Technology for having given us this opportunity to embark on this project.

We express our sincere and heartfelt thanks to **Dr. S. Thangasamy**, Dean, Department of Computer Science and Engineering, for his kind guidance and support.

We would like to express our sincere thanks to our project coordinator **Mrs. P. Devaki**, for her valuable guidance during the course of the project.

We would also like to thank our class advisor **Mrs. R. Kalaiselvi**. for her constant support and guidance.

We would like to thank our guide **Mrs. C. Ramathilagam**, without whose motivation and guidance we would not have been able to embark on a project of this magnitude. We express our sincere thanks for her valuable guidance, benevolent attitude and constant encouragement.

We reciprocate the kindness shown to us by the staff members of our college, people at home and our beloved friends who have contributed in the form of ideas, constructive criticism and encouragements for the successful completion of the project.

ABSTRACT

ABSTRACT

Nowadays, as information systems are more open to the Internet, the importance of secure networks is tremendously increased. New intelligent Intrusion Detection Systems (IDSs) based on sophisticated algorithms rather than current signature-based detections are in demand.

We propose a new data-mining based technique for intrusion detection using an ensemble of binary classifiers with feature selection and multiboosting simultaneously. Our model employs feature selection so that the classification using binary classifier for each type of attack can be more accurate, which improves the detection of attacks that occur rarely but are malicious in the training data.

Based on the accurate binary classifiers, our model applies a new ensemble approach which aggregates each binary classifier's decisions for the same input and decides which class is most suitable for a given input. During this process, the potential bias of certain binary classifier could be alleviated by other binary classifiers' decision. Bias refers to an error due to a learning algorithm while variance refers to an error due to the learned model.

Our model also makes use of multiboosting for reducing both variance and bias. The experimental results show that our approach provides better performance in terms of accuracy. Future works will extend our analysis to a new 'Protected Repository for the Defense of Infrastructure against Cyber Threats' (PREDICT) dataset as well as real network data.

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES

LIST OF ABBREVIATIONS

P-3107

LIST OF ABBREVIATIONS

1.	TCP	- Transmission Control Protocol
2.	UDP	- User Datagram Protocol
3.	ICMP	- Internet Control Message Protocol
4.	DOS	- Denial of Service
5.	R2L	- Remote to Local
6.	U2R	- User to Root

CONTENTS

TABLES OF CONTENTS

INTRODUCTION

INTRODUCTION

There has been a recent awareness of the risk associated with network attacks by criminals or terrorists, as information systems are now more open to the Internet than ever before. Records made available by the Pentagon showed that they logged over 79,000 attempted intrusions in 2005 with about 1,300 successful ones. The deployment of sophisticated firewalls or authentication systems is no longer enough for building a secure information system. In addition, most of intrusion detection systems nowadays rely on handcrafted signatures just like anti-viruses which have to be updated continuously in order to be effective against new attacks.

There is a need now to focus on the detection of unknown intrusions instead of relying on this signature based approach. This has led to another approach to intrusion detection which consists of detecting anomalies on the network. The anomaly detection attempts to quantify usual or acceptable behavior and flags other irregular behavior as potentially intrusive .

It is generally believed that intrusions show something which differs from the normal pattern, and that any unknown intrusion will present patterns more similar to known intrusion than to normal data. This observation suggests that intrusion detection could be considered as a data analysis problem. Additionally, by gathering network traffic, computing the right features and using the right classification algorithm, the system should be able to detect known intrusions as well as new intrusions.

In this paper, we will present a new data-mining based technique for intrusion detection using an ensemble of binary classifiers with feature selection and multiboosting simultaneously. Our model employs feature selection so that the binary classifier for each type of attack can be more accurate, which improves the detection of attacks that occur rarely but malicious in the training data. Based on the accurate binary classifiers, our model applies a new ensemble approach which aggregates each binary classifier's decisions for the same input and decides which class is most suitable for a given input. During this process, the potential bias of certain binary classifier could be alleviated by other binary classifiers' decision. Bias refers to an error due to a learning algorithm while variance refers to an error due to the learned model.Our model also makes use of multiboosting for reducing both variance and bias.

LITERARY SURVEY

1.1 PRELIMINARY INVESTIGATION

A. Dataset and Attack Types

The dataset we used in our experiments is the KDD '99 cup version of the 1998 'Defense Advanced Research Projects Agency' (DARPA) intrusion evaluation dataset by the MIT Lincoln Laboratory. The DARPA dataset is used as a test bed for intrusion-detection evaluation and was generated from raw TCP/IP dump data.We are aware that the KDD '99 cup dataset has been criticized because it is out-dated and artificially synthesized, and that there is a new 'Protected Repository for the Defense of Infrastructure against Cyber Threats' (PREDICT) dataset .

However, for the PREDICT dataset, we could not find other experimental results for performance comparison. Additionally, our approach has been less sensitive to a dataset, e.g., a uniformly sampled training data and no bias in design process (in terms of using data, arbitration, and multiboosting) so that it is expected to adapt well for other datasets. Thus, we decided to use the KDD '99 cup dataset in our experiments. The training dataset contains 24 attack types that could be classified into four main categories: Probing (Probe), Denial of Service (DOS), User to Root (U2R), and Remote to Local (R2L).

- **Intrinsic features:**

The features are common to any network connection, they are basic features like: duration of the connection, protocol type, service type, and total bytes sent.

- **Time-based feature:**

These are the "same host" and "same service" features which are called together time-based traffic features. These features, combined with basic features, could be used to detect DOS and fast probing attacks with the two second time window.

• **Host-based traffic features:**

We have the host-based traffic features which were constructed using a window of 100 connections to the same destination host instead of a time window. These features are useful for detecting slow probing attacks that require several minutes to execute.

• **Content-based features:**

Unlike DOS and Probe attacks which require many connections in a short amount of time, R2L and U2R attacks do not have any sequential patterns. Thus, the content features were computed by examining packet contents, e.g., computing the number of failed login could be helpful to detect a U2R attack. The main challenge in using the KDD cup dataset is the distribution of the attacks in the training and testing dataset.

Pattern-recognition and machine-learning techniques trained with the KDD cup training data and tested with their testing data failed to detect the majority of U2R and R2L attacks with an acceptable false positive rate.Some attacks which do not exist on the training set appear on the testing set, e.g., Mscan and Saint in Probe. Since all these attacks can be characterized by similar behavior in the feature space when it comes to relevant feature in a category, we will be able to detect some of them if we have enough training data for this category.

2.2 Proposed Model

Our model is described as follows:

For each trial *i*, *i*=1…*T*, where *T* is the total # of trials,

(1) A sample training set is generated by a multibooster using wagging.

(2) Binary classifiers are generated for each class of event using relevant features for the class and the C4.5 classification algorithm . Binary classifiers are derived from the training sample by considering all classes other than the current class as other, e.g., *Cnormal* will consider two classes: normal and other. The purpose of this phase is to select different features for different classes by applying the information gain or gain ratio in order to identify relevant features for each binary classifier. Moreover, applying the information gain or gain ratio will return all the features that contain more information for separating the current class from all other classes. The output of this ensemble of binary classifiers will be decided using arbitration function based on the confidence level of the output of individual binary classifiers.

(3) The ensemble classifier is used by the multibooster in order to calculate the classification error, and derive the next training set.

(4) After *T* trials, the final committee is formed and it will be used by our intrusion detection system.

2.2.3Feasibility Study

Feasibility study is the test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of recourses. It focuses on the evaluation of existing system and procedures analysis of alternative candidate system cost estimates. Feasibility analysis was done to determine whether the system would be feasible.

The development of a computer based system or a product is more likely plagued by resources and delivery dates. Feasibility study helps the analyst to decide whether or not to proceed, amend, postpone or cancel the project, particularly important when the project is large, complex and costly.

Once the analysis of the user requirement is complement, the system has to check for the compatibility and feasibility of the software package that is aimed at. An important outcome of the preliminary investigation is the determination that the system requested is feasible.

Technical Feasibility:

The technology used can be developed with the current equipments and has the technical capacity to hold the data required by the new system.

- This technology supports the modern trends of technology.
- Easily accessible, more secure technologies.

Technical feasibility on the existing system and to what extend it can support the proposed addition.

Operational Feasibility:

This proposed system can easily implemented, as this is based on JSP coding (JAVA) & HTML .The database created is with MySql server which is more secure and easy to handle. The resources that are required to implement/install these are available. The person of the organization already has enough exposure to computers. So the project is operationally feasible.

Economical Feasibility:

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action. This system is more economically feasible which assess the brain capacity with quick & online test. So it is economically a good project.

SYSTEM DESCRIPTION

3.1 SOFTWARE DESCRIPTION

3.1.1 Java

James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan convinced Java at Sun Microsystems, Inc. in 1991. It took 18 months to develop the first working version. This language was initially called "Oak" but was renamed "Java" in 1995.

Object oriented programming is the core of Java. All Java Programs are object oriented data. The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple and powerful
- Secure
- Portable
- Object-Oriented
- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High performance
- Distributed and Dynamic.

With the emergence of World Wide Web, java was propelled to the forefront of the computer language design, because the web demanded portable programs. Java is programmers language it is logically consistent. Java is not a language with training wheels. It is language for professional programmers.

3.1.2 J2EE :

Short for Java 2 Platform Enterprise Edition. J2EE is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitiered, Web-based applications.

The Java 2 Platform, Enterprise Edition (J2EE) is a set of coordinated specifications and practices that together enable solutions for developing, deploying , and managing multi-tier server-centric applications.

Some of the key features and services of J2EE:

- At the client tier, J2EE supports pure HTML, as well as Java applets or applications. It relies on Java Server Pages and servlet code to create HTML or other formatted data for the client.
- Java Database Connectivity (JDBC), which is the Java equivalent to ODBC, is the standard interface for Java databases.
- The Java servlet API enhances consistency for developers without requiring a graphical user interface.

JSP: Java Server Pages

JSP is a widely used general-purpose scripting language that is especially suited for Web development .In our proposed system JSP is the server side scripting language.

Java Server Pages or JSP for short is Sun's solution for developing dynamic web sites. JSP provide excellent server side scripting support for creating database driven web applications. JSP enable the developers to directly insert java code into jsp file, this makes the development process very simple and its maintenance also becomes very easy. JSP pages are efficient, it loads into the web servers memory on receiving the request very first time and the subsequent calls are served within a very short period of time.

Main reasons to use JSP:

- ❖ Multi platform.
- ❖ Component reuse by using JavaBeans and EJB.
- ❖ We can take one JSP file and move it to another platform, web server or JSP Servlet engine.

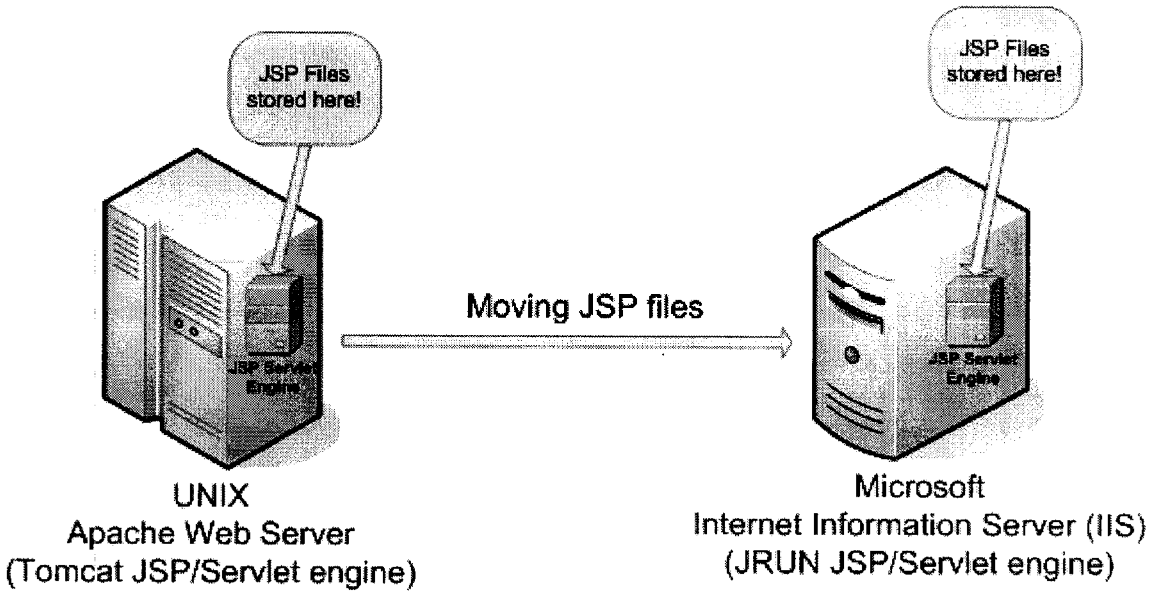


Fig 1. JSP Architecture

In today's environment most web sites servers dynamic pages based on user request. Database is very convenient way to store the data of users and other things. JDBC provide excellent database connectivity in heterogeneous database environment. Using JSP and JDBC its very easy to develop database driven web application.

Java is known for its characteristic of "write once, run anywhere." JSP pages are platform independent. Your port your .jsp pages to any platform.

The most significant of the many good reasons is that it is amazingly easy to develop sophisticated Web sites with JSPs. Through a mechanism called JavaBeans, JSPs have made it possible for large teams or individuals working on complex projects to divide the work in such a way as to make each piece simple and manageable, without sacrificing any power. JSPs also provide a great deal of flexibility when generating HTML, through the ability to create HTML-like custom tags.

JSPs are compiled into Java Servlets by a JSP compiler. A JSP compiler may generate a servlet in Java code that is then compiled by the Java compiler, or it may generate byte code for the servlet directly. "Java Server Pages" is a technology released by Sun.

In addition to this fundamental ease of development, high-quality JSP tools are readily available and easy to use. There is no need to buy expensive software or commit to a particular operating system in order to use JSPs.

JSP provides a Tag based approach to develop the server side executable application which is used to generate dynamic content.

There are five main tags:

1. Declaration tag
2. Expression tag
3. Directive tag
4. Scriptlet tag
5. Action tag

Declaration tag (<%! %>)

This tag allows the developer to declare variables or methods.

Before the declaration it must have <%!

At the end of the declaration, the developer must have %>

Code placed in this tag must end in a semicolon (;)

Expression tag (<%= %>)

This tag allows the developer to embed any java expression and is short for out.println() .

A semicolon(;) does not appear at the end of the code inside the tag.

Directive tag (<%@ directive ... %>)

A JSP directive gives special information about the page to the JSP Engine.

There are three main types of directives:

1) page - processing information for this page.
2) Include - files to be included.
3) Tag library - tag library to be used in this page.

Action tag

There are three main roles of action tags :

1) enable the use of server side Javabeans
2) transfer control between pages
3) browser independent support for applets.

Scriptlet tag (<% ... %>)

Between <% and %> tags,any valid Java code is called a Scriptlet. This code can access any variable or bean declared.

Problems with Servlet – If we want to generate a page, which consists of some dynamic content, then we have to include the entire static content into the Servlet (i.e, out.println method) along with the dynamic content. i.e.

when a huge view has to be generated then there was a problem with the application development and modifications(view/application).
While developing web applications using JSP technology we can separate the business logic from the presentation logic. Separation of presentation logic from business logic helps in reducing the time required to manage the project in future. Thus the advantage of JSP is that it provides a better mechanism to develop and maintain huge view compare to servlets.

3.1.3 WinPcap :

WinPcap is an open source library for packet capture and network analysis for the Win32 platforms.

Most networking applications access the network through widely used operating system primitives such as sockets. It is easy to access data on the network with this approach since the operating system copes with the low level details (protocol handling, packet reassembly, etc.) and provides a familiar interface that is similar to the one used to read and write files.

The purpose of WinPcap is to give this kind of access to Win32 applications; it provides facilities to:

- capture raw packets.
- transmit raw packets to the network .
- gather statistical information on the network traffic .

3.1.4 Jpcap

Jpcap is a Java class package that allows Java applications to capture and/or send packets to the network.

Jpcap is based on libpcap/winpcap and Raw Socket API. Therefore, Jpcap is supposed to work on any OS on which libpcap/winpcap has been implemented. Currently, Jpcap has been tested on FreeBSD 3.x, Linux RedHat 6.1, Fedora Core 4, Solaris, and Microsoft Windows 2000/XP.

3.1.5 Apache Tomcat :

Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies.

Apache Tomcat is a Java servlet engine and Java Server Pages processor which can run standalone or integrated with the Apache Web server to serve specific virtual paths. This is an aspect of the web that Apache's Tomcat is very good at because Tomcat provides both Java servlet and Java Server Pages (JSP) technologies (in addition to traditional static pages and external CGI programming). The result is that Tomcat is a good choice for use as a web server for many applications.

Apache Tomcat software requirements :--

1. Java 2 Software Development Kit version 1.2.2 or higher from Sun Microsystems, formerly known as the Java Development Kit or JDK.
2. The Apache Tomcat servlet container and JSP environment from the Jakarta Apache site.
3. The latest version of the Apache Web server for Windows binary distribution of the Apache server for versions 1.3.17.

Apache Tomcat is a Web Server implements Servlet and Java Server Pages specification from Java Software and includes many additional features that make it a useful platform for developing and deploying web applications and web services.

Tomcat will operate under any Java Development Kit (JDK) environment that provides a JDK 1.5 (also known as Java2 Standard Edition or J2SE) or later platform. The developer will need a Java Development Kit; as opposed to a Java Runtime Environment so that Servlet other classes and JSP pages can be complied. Tomcat 4 has been extensively tested with JDK 1.5, which is recommended Tomcat to set up a development environment, organize the source code, and then build and test the application.

3.2 HARDWARE DESCRIPTION

• Processor	: PentiumIII
• Clock speed	: 550MHz
• Hard Disk	: 20GB
• RAM	: 128MB
• Cache Memoy	: 512KB
• Operating System	: Windows2000
• Monitor	: Color Monitor
• Keyboard	: 104Keys
• Mouse	: 3Buttons

IMPLEMENTATION DETAILS

4. IMPLEMENTATION DETAILS

4.1 MODULES

1. Analyzing the Data set.
2. Processing the Dataset.
3. Data mining using binary classifier (C4 Algorithm).
4. Multi boosting.

4.2 MODULE DESCRIPTION

4.2.1 Analyzing the Data set:

A **data set** (or **dataset**) is a collection of data, usually presented in tabular form. Each column represents a particular variable. Each row corresponds to a given member of the data set in question. It lists values for each of the variables, such as height and weight of an object or values of random numbers. Each value is known as a datum. The data set may comprise data for one or more members, corresponding to the number of rows.

The values may be numbers, such as real numbers or integers, for example representing a person's height in centimeters, but may also be nominal data (i.e., not consisting of numerical values), for example representing a person's ethnicity. More generally, values may be of any of the kinds described as a level of measurement. For each variable, the values will normally all be of the same kind. However, there may also be "missing values", which need to be indicated in some way.

4.2.2 Processing the Dataset:

In this module we are going to receive the network packet and extract attributes using the WinPcap and JPCap.

In Information technology, a **packet** is a formatted unit of data carried by a packet mode computer network. Computer communications links that do not support packets, such as traditional point-to-point telecommunications links, simply transmit data as a series of bytes, characters, or bits alone. When data is formatted into packets, the bitrate of the communication medium can better be shared among users than if the network were circuit switched. By using packet switched networking it is also harder to guarantee a lowest possible bitrate.

A packet consists of two kinds of data: control information and user data (also known as *payload*). The control information provides data the network needs to deliver the user data, for example: source and destination addresses, error detection codes like checksums, and sequencing information. Typically, control information is found in packet headers and trailers, with user data in between.

Different communications protocols use different conventions for distinguishing between the elements and for formatting the data. In Binary Synchronous Transmission, the packet is formatted in 8-bit bytes, and special characters are used to delimit the different elements. Other protocols, like Ethernet, establish the start of the header and data elements by their location relative to the start of the packet. Some protocols format the information at a bit level instead of a byte level.

A good analogy is to consider a packet to be like a letter: the header is like the envelope, and the data area is whatever the person puts inside the envelope. A difference, however, is that some networks can break a larger packet into smaller packets when necessary (note that these smaller data elements are still formatted as packets).

A network design can achieve two major results by using packets: *error detection* and *multiple host addressing.*

A nominal traffic profile consists of single and joint distributions of various packet attributes that are considered unique for a site. Candidate packet attributes from IP headers are:

1. packet size,
2. Time-to-Live (TTL) values,
3. protocol-type values, and
4. source IP prefixes.

Those from TCP headers are:

5. TCP flag patterns and
6. server port numbers, i.e., the smaller of the source port number and the destination port number.

Server port number is more stable than sort/destination port numbers because most of the well-known port numbers are small numbers (e.g., below 1,024) and a large portion of Internet traffic uses the well-known port numbers. To increase the number of attributes, we can employ joint distributions of the fraction of packets having various combinations, such as:

7. <packet-size and protocol-type>,
8. <server port number and protocol-type>, and
9. <source IP prefix, TCP flags and packet size>, etc.

Joint distributions often better represent the uniqueness of the traffic distribution for a site, and are harder to guess for the attackers. As many different combinations of single attributes as needed may be used while the storage space permits.

4.2.3 Data mining using binary classifier (C4 Algorithm):

Binary classifiers are generated for each class of event using relevant features for the class and classification algorithm .Binary classifiers are derived from the training sample by considering all classes other than the current class as other, e.g., Cnormal will consider two classes: normal and other. The purpose of this phase is to select different features for different classes by applying the information gain or gain ratio in order to identify relevant features for each binary classifier.

Moreover, applying the information gain or gain ratio will return all the features that contain more information for separating the current class from all other classes. The output of this ensemble of binary classifiers will be decided using arbitration function based on the confidence level of the output of individual binary classifiers

Multi boosting

The effect of combining different classifiers can be explained with the theory of bias-variance decomposition. Bias refers to an error due to a learning algorithm while variance refers to an error due to the learned model. The total expected error of a classifier is the sum of the bias and the variance. In order to reduce bias and variation, some ensemble approaches have been introduced: Adaptive Boosting (AdaBoost) ,Bootstrap Aggregating (Bagging),Wagging and Multiboosting. This is why the idea emerged of combining both in order to profit from the advantages of both algorithms and obtain an overall error reduction

4.3 **Data Flow Diagram**

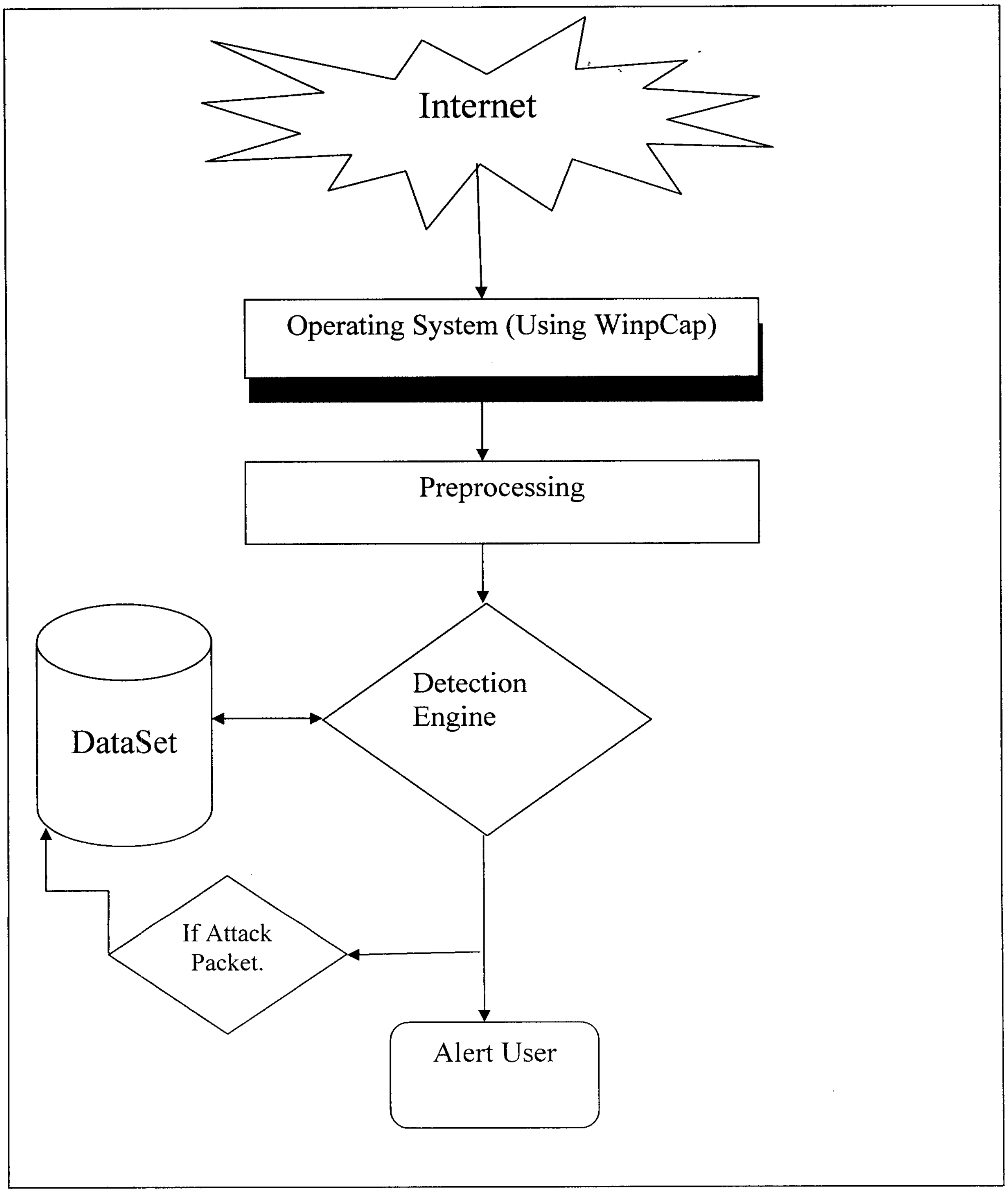


Fig 2. Data Flow Diagram

5. DESIGN AND DEVELOPMENT PHASE

5.1 DESIGN AND DEVELOPMENT

The design phase is a multi step process which focuses on system creation with the help of user specifications and information gathered in the above phases. It is the phase where the system requirements are translated to operational details.

5.2 SYSTEM DESIGN

System Design is the process of making the newly designed system fully operational and consistent in performance. The following steps have been followed in the implementation of the system.

- Implementation in planning
- User Training

As the part of implementation, the system is taken the site and Loaded on to client's computer.Some of the user's level, exposure to computer etc. A detailed documentation is prepared for the employees and they trained to access the software. These users are trained first and they can run the system for a month.

After installation of software, the hardware specifications are checked. If hardware specifications are satisfactory, then the software is loaded for pilot run. User training starts at this time itself. Users will be given a user manual, which documents how to use the system and all the exception handling procedures.

5.3 INPUT DESIGN

Input design is the part of overall system design which requires very careful attention. Often the collection of input data is the most expensive part of the system, in terms of both the equipment used and the number of people involved; it is the point of most contact for the users with the computer system; and it is prone to error. If data going into the system are incorrect, then the processing and output will magnify these error.

In this system inputs are given in two ways, the Existing users can directly enter into the system using login form, and new users have to register all their details in the registration form provided.

Input design is the very important part in the project and should be concentrated well as it is prone to error. The data that are to be inserted are to be inserted with care as this plays a very important role. In order to get the meaningful output and to achieve good accuracy the input should be acceptable and understandable by the user.

5.4 OUTPUT DESIGN

Output design plays a very important role in a system. Getting a correct output is a task that has to be concentrated, as a system is validated as a correct one only if it gives the correct output according to the input.

5.5 USE CASE diagram

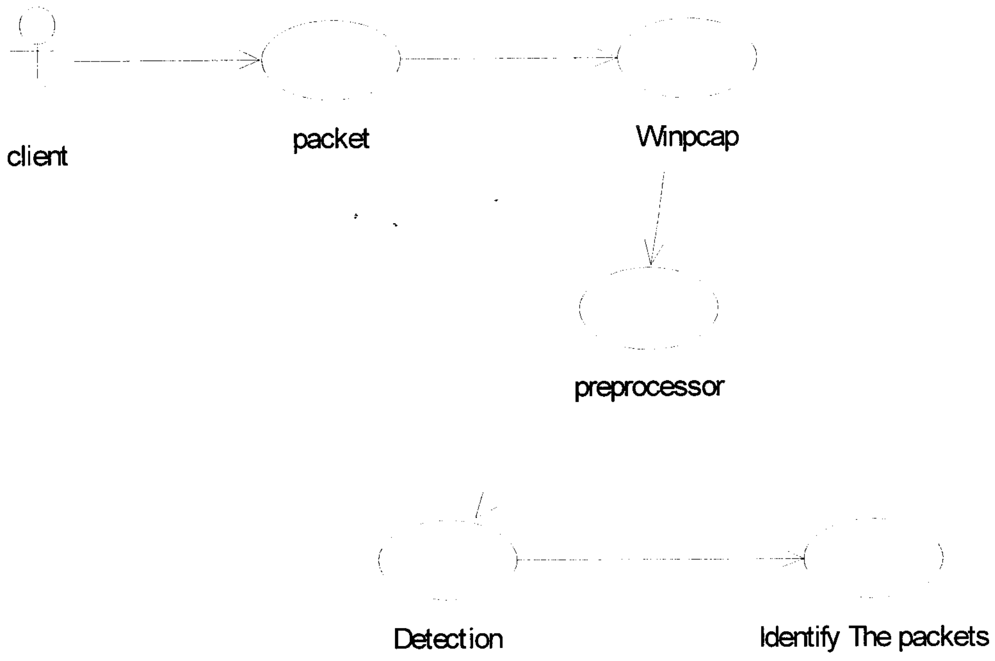


Fig 3 .Use Case Diagram

5.6 CLASS diagram :

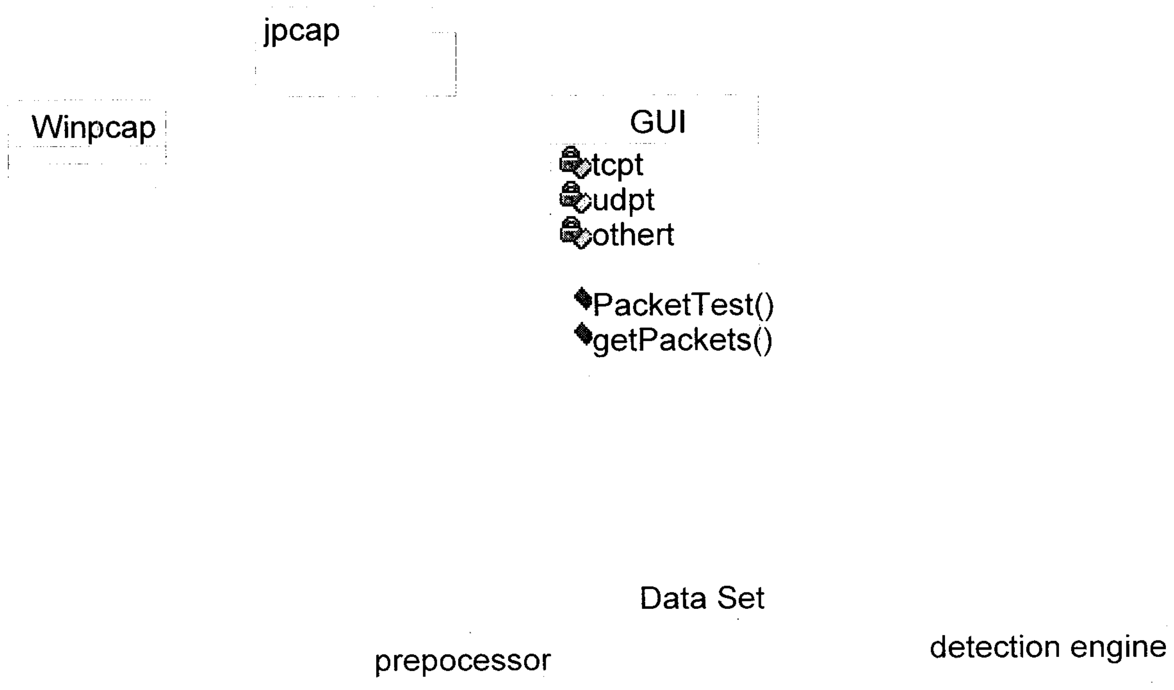


Fig 4: Class Diagram

5.7 ER diagram :

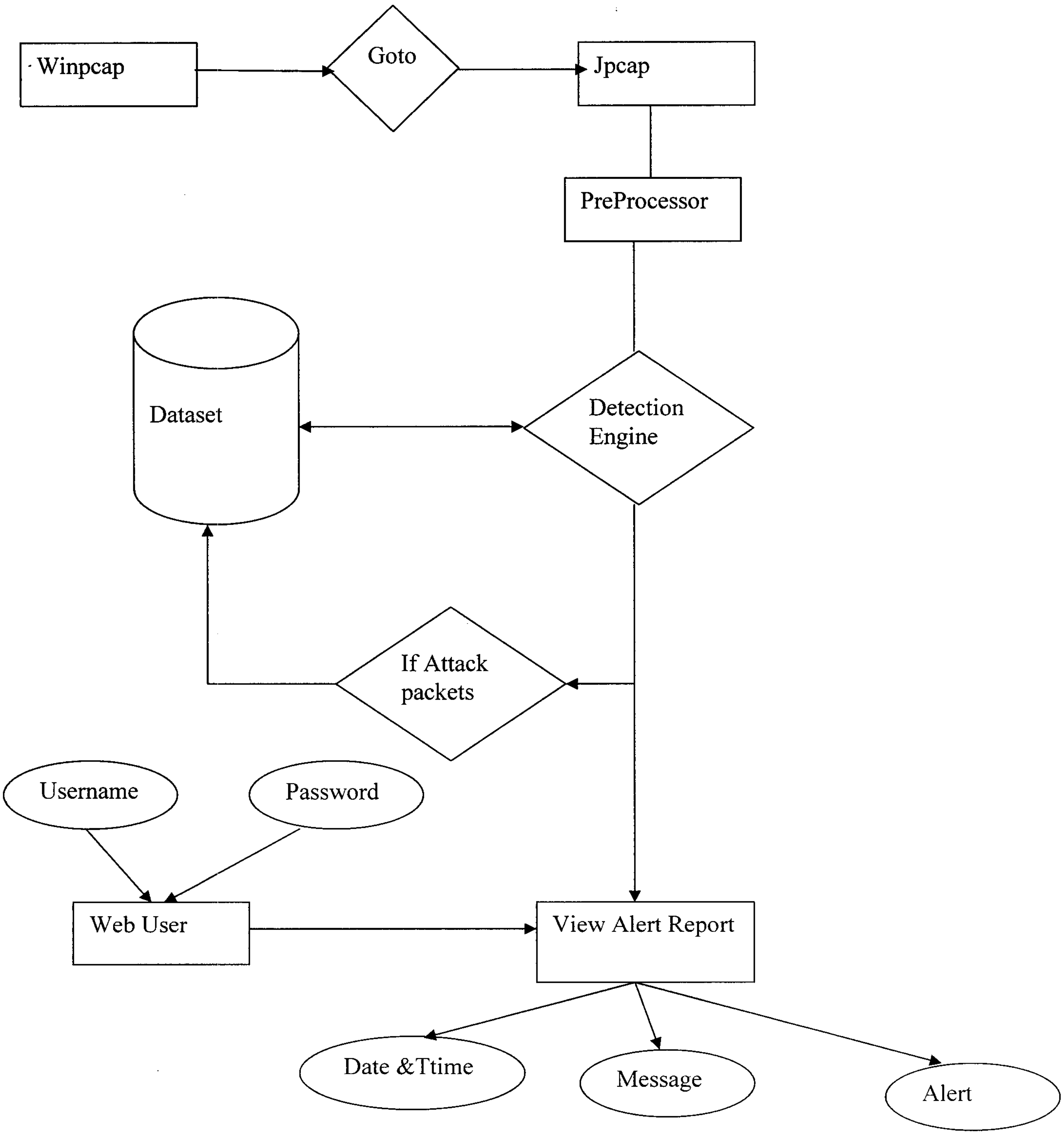


Fig 5 :ER Diagram

5.8 DATABASE DESIGN

A well-designed database is essential for the performance of the system. Several tables are manipulated for varying purposes.

Login :

Table Name	:	Login
Purpose	:	Provides authenticated login.
Primary Key	:	Password

SNo	Field Name	Type	Width	Constraint	Description
1	Username	varchar	50	Not null	Unique name given by the user
2	Password	varchar	50	Primary key	Unique password

Table 5.1- Login table

Alert Log :

Table Name	:	Alert Log
Purpose	:	Allows to view the report generated.
Primary Key	:	Message

Sl No	Field Name	Type	Width	Constraint	Description
1	Date	varchar	50	Not null	The date when the test is performed
2	Message	varchar	50	Primary key	Type of the intrusion found
3	Action	varchar	50	Not null	Alert message

Table 5.2- Alert Log table

5.9 SYSTEM TESTING & TEST PLANS

Testing techniques and Testing strategies :

There are four testing strategies that are mainly used. These are,

- Unit Testing
- Integration Testing
- Validation Testing
- System Testing

This system was tested using Unit Testing and Integration Testing strategies because these were the most relevant approaches for this project.

Software Testing Techniques:

Specification-based testing:

Specification-based testing aims to test the functionality of software according to the applicable requirements. Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Specification-based testing is necessary, but it is insufficient to guard against certain risks.

A software engineering product can be tested in one of the two ways:

- Black Box Testing
- White Box Testing

5.8.1 Black Box Testing:

Knowing a specified function that a product has been designed to perform, determine whether each function is fully operational. Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing. Tests are based on requirements and functionality.

5.8.2 White Box Testing:

Knowing the internal workings of a software product determine whether the internal operations implementing the functions performs according to the specification, and all the internal components have been adequately exercised.

The following types of white box testing exist:

- API testing (application programming interface) - Testing of the application using Public and Private APIs
- Code coverage - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
- Fault injection methods - improving the coverage of a test by introducing faults to test code paths
- Mutation testing methods
- Static testing - White box testing includes all static testing

5.8.3 Unit testing:

We adopt white box testing when using this testing technique. This testing was carried out on individual components of the software that were designed. Each individual module was tested using this technique during the coding phase. Every component was checked to make sure that they adhere strictly to the specifications spelt out in the Data Flow Diagram and ensure that they perform the purpose intended for them.

All the names of the variables are scrutinized to make sure that they truly reflect the element they represent. All the looping mechanisms were verified to ensure that they were as decided. Besides these, we trace through the code manually to capture syntax errors and logical errors.

5.8.4 Integration Testing:

After finishing the Unit Testing, next is the integration testing process. In this testing process we put our focus on identifying the interfaces between components and their functionality as dictated in the Data Flow Diagram. The Bottom Up incremental approach was adopted during these testing. Low-level modules are integrated and combined as a cluster before testing.

The Black Box testing technique was employed here. The interfaces between the components were tested first. This allowed identifying any wrong linkages or parameters passing early in the development process as it can be just passed in a set of data and checked if the result returned is an accepted one.

5.8.5 Validation Testing:

Software testing and validation is achieved through a series of black box test cases. A test procedure defines specific test cases that demonstrate conformity with the requirements. Both, the plan the procedure are designed to ensure that all functional requirements are achieved, documentation is correct and other requirements are met. After each validation test case has been conducted, one of the two possible conditions exists. They are,

- The function or performance characteristics confirm to specification and are accepted.
- A deviation from the specification is uncovered and a deficiency list is created.

The deviation or error discovered at this stage in project can rarely be corrected prior to scheduled completion. It is necessary to negotiate with the customer to establish a method of resolving deficiencies.

5.8.6 System Testing

System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all the work should verify that all system elements have been properly integrated and perform integrated functions.

System testing also ensures that the project works well in the environment. It traps the errors and allows convenient processing of errors without coming out of the program abruptly.

Recovery testing is done in such a way that failure is forced to a software system and checked whether the recovery is proper and accurate. The performance of system is highly effective.

5.8.7 Web Testing

Our proposed project is of 3 tier applications (developed for Internet). Here we will be having Browser, web server and DB server. The applications accessible in browser would be developed in HTML, JavaScript etc. We can monitor through these applications for the web server would be developed in Java, JSP, JavaScript, (All the manipulations are done on the web server with the help of these programs developed).The DBserver would be having Oracle, Sql server, Sybase, Mysql etc. (All data is stored in the database available on the DB server).

The tests performed on these types of applications would be
- User interface testing
- Functionality testing
- Security testing
- Browser compatibility testing
- Load / stress testing
- Storage and data volume testing

A web-application is a three-tier application.This has a browser (monitors data) [monitoring is done using html-> webserver (manipulates data) [manipulations are done using programming languages or scripts like adv java,jsp, javascript,] -> database server (stores data) [data storage and retrieval is done using databases like mysql].The types of tests, which can be applied on this type of applications, are:

1. User interface testing for validation & user friendliness
2. Functionality testing to validate behaviors, i/p, error handling, o/p, manipulations, services levels, order of functionality, links, content of web page & backend coverage's

3. Security testing
4. Browser compatibility
5. Load / stress testing
6. Interoperability testing
7. Storage & data volume testing

CONCLUSION

5. CONCLUSION

In this paper, we propose a new data-mining based approach by combining multiboosting and an ensemble of binary classifiers with feature selection using either the information gain or the gain ratio criterion. This approach consists of three major functions:

1) Generation of accurate binary classifiers by applying different features for different types of attacks,

2) A new ensemble approach of the binary classifiers for removing bias,

3) Applying multiboosting for reducing both bias and variance. This model performs well and we even obtain 93.8128% detection rate using the gain ratio criterion as well as high detection rates for U2R and R2L compared to other works.

The proposed system performs better than the winning entry of the KDD cup in term of accuracy and cost. Our experiments are solely based on the KDD/DARPA data set so that we need to verify our result in other network environments. Future works will extend the analysis to the PREDICT dataset and real network data.

APPENDIX

APPENDIX

7.1 SAMPLE CODE

Main.java :

```
// calls other modules and initiates packet capturing

package c45;

import c45.algorithm.Classifiers;
import c45.preprocess.Instances;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.util.Hashtable;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import c45.packet.*;
import java.sql.SQLException;

public class Main {
   Instances  m_Instances;
   Classifiers  dd ;
  public static Map packetdata ;

  Main()
  {
   dd = new Classifiers();
   packetdata = new Hashtable();
}

   public static void main(String[] args) {
      try {
          Main main = new Main();
          main.read();
         // Main.packetProcess();
          // TODO code application logic here
      } catch (IOException ex) {
          Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null,
ex);
      }
```

```
        // TODO code application logic here
    }

    void read () throws IOException{
        Reader r = null;
        try {
            String url = "c:\\dataset1.arff";
            File file = new File(url);
            r = new BufferedReader(new FileReader(file));
            setInstances(new Instances(r));
            dd.startClassifier(getInstances());
             Runnable r1 = new Runnable() {

                            public void run() {
                           System.out.println("dsf");
                                  PacketSniffer sniffer = new PacketSniffer();
                             System.out.println("dsf");
                try {
                    sniffer.startCapture();
                     System.out.println("dsf");
                } catch (SQLException ex) {

Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
                }
                       }
                };
                Thread th1 = new Thread(r1);
                th1.start();
        } catch (IOException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null,
ex);
        } finally {
            r.close();
        }
    }


   public void setInstances(Instances inst) {
    m_Instances = inst;
    System.out.println(m_Instances.relationName());
    System.out.println(m_Instances.numInstances());
   }
```

```
  // Gets the working set of instances.

 public Instances getInstances() {

   return m_Instances;
  }
}
```

Packet Sniffer.java:

```
// It captures the incoming packets and extracts data from them

package c45.packet;

import c45.database.*;
import c45.algorithm.binary.*;
import java.io.*;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import jpcap.*;
import jpcap.packet.*;


public class PacketSniffer {

//implements Runnable {

   JpcapCaptor jpcap;
   FileWriter filewriter;
   NetworkInterface[] devices = JpcapCaptor.getDeviceList();
   TCPAnalyzer tcp = new TCPAnalyzer();
   UDPAnalyzer udp = new UDPAnalyzer();
   Calculation cal = new Calculation();
   Database db;


  // public static boolean ATTACK_DETECTED = false;
   public void startCapture() throws SQLException {
      try {
         try {
         db = new Database();
```

```
        } catch (ClassNotFoundException ex) {

Logger.getLogger(PacketSniffer.class.getName()).log(Level.SEVERE, null,
ex);
        }
        jpcap = JpcapCaptor.openDevice(devices[1], 65535, true, 20);

      // PacketAnalyzer analyzer = new PacketAnalyzer();

        while (!Thread.currentThread().isInterrupted()) {

            Packet packet = jpcap.getPacket();

          if (packet == null) {
              try {
                  Thread.sleep(10);
              } catch (InterruptedException e) {
                  Thread.currentThread().interrupt();
                  continue;
              }
          } else {
              System.out.println("Packets Details" + packet.toString() );

              if (packet instanceof IPPacket) {
                  IPPacket ip = (IPPacket) packet;
              }if (packet instanceof TCPPacket){
                  tcp.analyze(packet);
                  //System.out.println("asa----------------
"+tcp.getValue("Source Port"));
                  //    System.out.println("asa----------------
"+tcp.getValue("Sequence Number"));
                  //    System.out.println("asa----------------
"+tcp.getValue("Window Size"));
                  tcp.getValues();
              String result =  cal.ReadRule(tcp.getValue());
              if(result.equals("normal")){}
              else{

    File f=new File("c:\\dataset1.arff");
    FileOutputStream fop=new FileOutputStream(f,true);
    if(f.exists()){

        fop.write((tcp.getAttribute()+result+"\n").getBytes());
fop.flush();
fop.close();
System.out.println("The data has been written");
}

            System.out.println(tcp.getAttribute()+result);

             db.executeUpdate1(result);
        }
        System.out.println("result"+result);
        }if(packet instanceof UDPPacket){
           udp.analyze(packet);
           udp.getValues1();
          String result =  cal.ReadRule(udp.getValue());

        if(result.equals("normal")){}
        else{
             File f=new File("c:\\dataset1.arff");
             FileOutputStream fop=new FileOutputStream(f,true);
             if(f.exists()){
                fop.write(
                      (udp.getAttribute()+result+"\n").getBytes());
               //  fop.write("\n".getBytes());
               fop.flush();
               fop.close();
               System.out.println("The data has been written");
             }else{
                 System.out.println("This file is not exist");
             }

            // filewriter = new FileWriter("c:\\dataset1.txt",true);
             System.out.println(udp.getAttribute()+result);


            db.executeUpdate1(result);
        }
          System.out.println("result"+result);
        }
      }
   }
} catch (IOException e) {
   e.printStackTrace();
}
```

```
  }

public static void main(String arg[]) throws SQLException{
   PacketSniffer packetsniffer =  new PacketSniffer();
   packetsniffer.startCapture();
  }
}
```

Bin c45 model selection.java:

// initiates the split of the data set into different classes based upon the model

```
package c45.algorithm.binary;

import c45.preprocess.Attribute;
import c45.preprocess.Instances;
import c45.preprocess.Utils;
import java.util.*;


// Class for selecting a C4.5-like binary (!) split for a given //dataset.

public class BinC45ModelSelection extends ModelSelection{

  /** Minimum number of instances in interval. */
  private int m_minNoObj;

  /** The FULL training dataset. */
  private Instances m_allData;

  public BinC45ModelSelection(int minNoObj,Instances allData){
    m_minNoObj = minNoObj;
    m_allData = allData;
  }

  // Sets reference to training data to null.

  public void cleanup() {

    m_allData = null;
  }

  // Selects C4.5-type split for the given dataset.
```

```
public final ClassifierSplitModel selectModel(Instances data){

 double minResult;
 double currentResult;
 BinC45Split [] currentModel;
 BinC45Split bestModel = null;
 NoSplit noSplitModel = null;
 double averageInfoGain = 0;
 int validModels = 0;
 boolean multiVal = true;
 Distribution checkDistribution;
 double sumOfWeights;
 int i;

 try{

  // Check if all Instances belong to one class or if not
  // enough Instances to split.
  checkDistribution = new Distribution(data);
  noSplitModel = new NoSplit(checkDistribution);
  if (Utils.sm(checkDistribution.total(),2*m_minNoObj) ||
      Utils.eq(checkDistribution.total(),
               checkDistribution.perClass(checkDistribution.maxClass())))
    return noSplitModel;

  // Check if all attributes are nominal and have a
  // lot of values.
  Enumeration enu = data.enumerateAttributes();
  while (enu.hasMoreElements()) {
    Attribute attribute = (Attribute) enu.nextElement();
    if ((attribute.isNumeric()) ||
      (Utils.sm((double)attribute.numValues(),
              (0.3*(double)m_allData.numInstances())))){
     multiVal = false;
     break;
    }
  }
  currentModel = new BinC45Split[data.numAttributes()];
  sumOfWeights = data.sumOfWeights();

  System.out.print("data.sumOfWeights();"+data.sumOfWeights());
```

```
// For each attribute.
for (i = 0; i < data.numAttributes(); i++){

  // Apart from class attribute.
  if (i != (data).classIndex()){

    // Get models for current attribute.
    currentModel[i] = new BinC45Split(i,m_minNoObj,sumOfWeights);
    currentModel[i].buildClassifier(data);

    // Check if useful split for current attribute
    // exists and check for enumerated attributes with
    // a lot of values.
    if (currentModel[i].checkModel())
      if ((data.attribute(i).isNumeric()) ||
          (multiVal || Utils.sm((double)data.attribute(i).numValues(),
                        (0.3*(double)m_allData.numInstances())))){
        averageInfoGain = averageInfoGain+currentModel[i].infoGain();
        validModels++;
      }
  }else
    currentModel[i] = null;
}

// Check if any useful split was found.
if (validModels == 0)
  return noSplitModel;
averageInfoGain = averageInfoGain/(double)validModels;

// Find "best" attribute to split on.
minResult = 0;
for (i=0;i<data.numAttributes();i++){
  if ((i != (data).classIndex()) &&
     (currentModel[i].checkModel()))


  if ((currentModel[i].infoGain() >= (averageInfoGain-1E-3)) &&
      Utils.gr(currentModel[i].gainRatio(),minResult)){
    bestModel = currentModel[i];
    minResult = currentModel[i].gainRatio();
   }
}
```

```
    // Check if useful split was found.
    if (Utils.eq(minResult,0))
       return noSplitModel;

    // Add all Instances with unknown values for the corresponding
    // attribute to the distribution for the model, so that
    // the complete distribution is stored with the model.
    bestModel.distribution().
       addInstWithUnknown(data,bestModel.attIndex());


    bestModel.setSplitPoint(m_allData);
    return bestModel;
   }catch(Exception e){
    e.printStackTrace();
   }
   return null;
  }

 // Selects C4.5-type split for the given dataset.

 public final ClassifierSplitModel selectModel(Instances train, Instances
test) {

   return selectModel(train);
  }
```

BinC45split.java:

```
// splits the dataset into various classes

package c45.algorithm.binary;

import c45.preprocess.Instance;
import c45.preprocess.Instances;
import c45.preprocess.Utils;
import java.util.*;

public class BinC45Split extends ClassifierSplitModel{

  /** Attribute to split on. */
  private int m_attIndex;
```

```
/** Minimum number of objects in a split.   */
private int m_minNoObj;

/** Value of split point. */
private double m_splitPoint;

/** InfoGain of split. */
private double m_infoGain;

/** GainRatio of split.  */
private double m_gainRatio;

/** The sum of the weights of the instances. */
private double m_sumOfWeights;

/** Static reference to splitting criterion. */
private static InfoGainSplitCrit m_infoGainCrit = new InfoGainSplitCrit();

/** Static reference to splitting criterion. */
private static GainRatioSplitCrit m_gainRatioCrit = new
GainRatioSplitCrit();

// Initializes the split model.

public BinC45Split(int attIndex,int minNoObj,double sumOfWeights){

  // Get index of attribute to split on.
  m_attIndex = attIndex;

  // Set minimum number of objects.
  m_minNoObj = minNoObj;

  // Set sum of weights;
  m_sumOfWeights = sumOfWeights;
}

// Creates a C4.5-type split on the given data.

public void buildClassifier(Instances trainInstances)
    throws Exception {

  // Initialize the remaining instance variables.
  m_numSubsets = 0;
  m_splitPoint = Double.MAX_VALUE;
  m_infoGain = 0;
  m_gainRatio = 0;

  // Different treatment for enumerated and numeric
  // attributes.
  if (trainInstances.attribute(m_attIndex).isNominal()){
   handleEnumeratedAttribute(trainInstances);
  }else{
   trainInstances.sort(trainInstances.attribute(m_attIndex));
   handleNumericAttribute(trainInstances);
  }
}

// Returns index of attribute for which split was generated.

 public final int attIndex(){

  return m_attIndex;
}

//  Returns gain ratio for the generated split.

public final double gainRatio(){
  return m_gainRatio;
}

public final double classProb(int classIndex,Instance instance,
                                        int theSubset) throws Exception {

  if (theSubset <= -1) {
   double [] weights = weights(instance);
   if (weights == null) {
      return m_distribution.prob(classIndex);
   } else {
      double prob = 0;
      for (int i = 0; i < weights.length; i++) {
        prob += weights[i] * m_distribution.prob(classIndex, i);
      }
      return prob;
   }
  } else {
   if (Utils.gr(m_distribution.perBag(theSubset), 0)) {
      return m_distribution.prob(classIndex, theSubset);
    } else {
      return m_distribution.prob(classIndex);
    }
  }
}

// Creates split on enumerated attribute.

  private void handleEnumeratedAttribute(Instances trainInstances)
    throws Exception {

  Distribution newDistribution,secondDistribution;
  int numAttValues;
  double currIG,currGR;
  Instance instance;
  int i;

  numAttValues = trainInstances.attribute(m_attIndex).numValues();
  newDistribution = new Distribution(numAttValues,
                                     trainInstances.numClasses());

  // Only Instances with known values are relevant.

  Enumeration enu = trainInstances.enumerateInstances();
  while (enu.hasMoreElements()) {
    instance = (Instance) enu.nextElement();
    if (!instance.isMissing(m_attIndex))
      newDistribution.add((int)instance.value(m_attIndex),instance);
  }
  m_distribution = newDistribution;

  // For all values
  for (i = 0; i < numAttValues; i++){

    if (Utils.grOrEq(newDistribution.perBag(i),m_minNoObj)){
      secondDistribution = new Distribution(newDistribution,i);

      // Check if minimum number of Instances in the two
      // subsets.
      if (secondDistribution.check(m_minNoObj)){
        m_numSubsets = 2;
        currIG = m_infoGainCrit.splitCritValue(secondDistribution,
                                        m_sumOfWeights);
        currGR = m_gainRatioCrit.splitCritValue(secondDistribution,
                                          m_sumOfWeights,
                                          currIG);
        if ((i == 0) || Utils.gr(currGR,m_gainRatio)){
          m_gainRatio = currGR;
          m_infoGain = currIG;
          m_splitPoint = (double)i;
          m_distribution = secondDistribution;
        }
      }
    }
  }
}

// Creates split on numeric attribute.

private void handleNumericAttribute(Instances trainInstances)
    throws Exception {

  int firstMiss;
  int next = 1;
  int last = 0;
  int index = 0;
  int splitIndex = -1;
  double currentInfoGain;
  double defaultEnt;
  double minSplit;
  Instance instance;
  int i;

  // Current attribute is a numeric attribute.
  m_distribution = new Distribution(2,trainInstances.numClasses());

  // Only Instances with known values are relevant.
  Enumeration enu = trainInstances.enumerateInstances();
  i = 0;
  while (enu.hasMoreElements()) {
    instance = (Instance) enu.nextElement();
    if (instance.isMissing(m_attIndex))
       break;
    m_distribution.add(1,instance);
    i++;
```

```
}
firstMiss = i;

// Compute minimum number of Instances required in each
// subset.
minSplit =  0.1*(m_distribution.total())/
  ((double)trainInstances.numClasses());
if (Utils.smOrEq(minSplit,m_minNoObj))
  minSplit = m_minNoObj;
else
  if (Utils.gr(minSplit,25))
      minSplit = 25;

// Enough Instances with known values?
if (Utils.sm((double)firstMiss,2*minSplit))
  return;

// Compute values of criteria for all possible split
// indices.
defaultEnt = m_infoGainCrit.oldEnt(m_distribution);
while (next < firstMiss){

  if (trainInstances.instance(next-1).value(m_attIndex)+1e-5 <
      trainInstances.instance(next).value(m_attIndex)){

    // Move class values for all Instances up to next
    // possible split point.
    m_distribution.shiftRange(1,0,trainInstances,last,next);

    // Check if enough Instances in each subset and compute
    // values for criteria.
    if (Utils.grOrEq(m_distribution.perBag(0),minSplit) &&
        Utils.grOrEq(m_distribution.perBag(1),minSplit)){
      currentInfoGain = m_infoGainCrit.
        splitCritValue(m_distribution,m_sumOfWeights,
                       defaultEnt);
      if (Utils.gr(currentInfoGain,m_infoGain)){
        m_infoGain = currentInfoGain;
        splitIndex = next-1;
      }
      index++;
    }
    last = next;
```

```
     }
     next++;
   }

   // Was there any useful split?
   if (index == 0)
     return;

   // Compute modified information gain for best split.
   m_infoGain = m_infoGain-(Utils.log2(index)/m_sumOfWeights);
   if (Utils.smOrEq(m_infoGain,0))
     return;

   // Set instance variables' values to values for
   // best split.
   m_numSubsets = 2;
   m_splitPoint =
     (trainInstances.instance(splitIndex+1).value(m_attIndex)+
      trainInstances.instance(splitIndex).value(m_attIndex))/2;

     // In case we have a numerical precision problem we need to choose
     // the smaller value
   if (m_splitPoint == trainInstances.instance(splitIndex +
1).value(m_attIndex)) {
     m_splitPoint = trainInstances.instance(splitIndex).value(m_attIndex);
   }

   // Restore distributioN for best split.
   m_distribution = new Distribution(2,trainInstances.numClasses());
   m_distribution.addRange(0,trainInstances,0,splitIndex+1);
   m_distribution.addRange(1,trainInstances,splitIndex+1,firstMiss);

   // Compute modified gain ratio for best split.
   m_gainRatio = m_gainRatioCrit.
     splitCritValue(m_distribution,m_sumOfWeights,
                    m_infoGain);
 }

 // Returns (C4.5-type) information gain for the generated split.

 public final double infoGain(){

   return m_infoGain;
```

```
}

// Prints left side of condition..

  public final String leftSide(Instances data){

 return data.attribute(m_attIndex).name();
}

// Prints the condition satisfied by instances in a subset.
 public final String rightSide(int index,Instances data){

  StringBuffer text;

  text = new StringBuffer();
  if (data.attribute(m_attIndex).isNominal()){
   if (index == 0)
      text.append(" = "+
                 data.attribute(m_attIndex).value((int)m_splitPoint));
   else
      text.append(" != "+
                 data.attribute(m_attIndex).value((int)m_splitPoint));
  }else
   if (index == 0)
      text.append(" <= "+m_splitPoint);
   else
      text.append(" > "+m_splitPoint);

  return text.toString();
}

 // Returns a string containing java source code equivalent to the          //
 test made at this node. The instance being tested is called "i".



public final String sourceExpression(int index, Instances data) {

  StringBuffer expr = null;
  if (index < 0) {
   return "i[" + m_attIndex + "] == null";
  }
  if (data.attribute(m_attIndex).isNominal()) {
   if (index == 0) {
      expr = new StringBuffer("i[");
   } else {
      expr = new StringBuffer("!i[");
   }
   expr.append(m_attIndex).append("]");
   expr.append(".equals(\"").append(data.attribute(m_attIndex)
                                   .value((int)m_splitPoint)).append("\")");
  } else {
   expr = new StringBuffer("((Double) i[");
   expr.append(m_attIndex).append("])");
   if (index == 0) {
      expr.append(".doubleValue() <= ").append(m_splitPoint);
   } else {
      expr.append(".doubleValue() > ").append(m_splitPoint);
   }
  }
  return expr.toString();
 }

 // Sets split point to greatest value in given data smaller or equal    //to old
 split point.

 public final void setSplitPoint(Instances allInstances){

  double newSplitPoint = -Double.MAX_VALUE;
  double tempValue;
  Instance instance;

  if ((!allInstances.attribute(m_attIndex).isNominal()) &&
      (m_numSubsets > 1)){
   Enumeration enu = allInstances.enumerateInstances();
   while (enu.hasMoreElements()) {
      instance = (Instance) enu.nextElement();
      if (!instance.isMissing(m_attIndex)){
        tempValue = instance.value(m_attIndex);
        if (Utils.gr(tempValue,newSplitPoint) &&
           Utils.smOrEq(tempValue,m_splitPoint))
         newSplitPoint = tempValue;
      }
   }
   m_splitPoint = newSplitPoint;
  }
```

```
}

// Sets distribution associated with model.

public void resetDistribution(Instances data) throws Exception {

 Instances insts = new Instances(data, data.numInstances());
 for (int i = 0; i < data.numInstances(); i++) {
  if (whichSubset(data.instance(i)) > -1) {
     insts.add(data.instance(i));
  }
 }
 Distribution newD = new Distribution(insts, this);
 newD.addInstWithUnknown(data, m_attIndex);
 m_distribution = newD;
}

// Returns weights if instance is assigned to more than one subset.
//  Returns null if instance is only assigned to one subset.

public final double [] weights(Instance instance){

 double [] weights;
 int i;

 if (instance.isMissing(m_attIndex)){
  weights = new double [m_numSubsets];
  for (i=0;i<m_numSubsets;i++)
     weights [i] = m_distribution.perBag(i)/m_distribution.total();
  return weights;
 }else{
  return null;
 }
}

// Returns index of subset instance is assigned to.
// Returns -1 if instance is assigned to more than one subset.

public final int whichSubset(Instance instance) throws Exception {

 if (instance.isMissing(m_attIndex))
  return -1;
 else{
      if (instance.attribute(m_attIndex).isNominal()){
        if ((int)m_splitPoint == (int)instance.value(m_attIndex))
          return 0;
        else
          return 1;
      }else
        if (Utils.smOrEq(instance.value(m_attIndex),m_splitPoint))
          return 0;
        else
          return 1;
    }
  }
}
```

C4.java:

```
// calculates the information gain and decides the attribute

package c45.algorithm.binary;


import c45.algorithm.Classifier;
import c45.preprocess.Instance;
import c45.preprocess.Instances;
import c45.preprocess.Utils;
import java.util.*;

// Class for generating an unpruned or a pruned C4.5 decision tree.


public class C4 extends Classifier { //implements OptionHandler {

  // To maintain the same version number after adding m_ClassAttribute
  static final long serialVersionUID = -217733168393644444L;

  // The decision tree
  private ClassifierTree m_root;

  // Unpruned tree?
  private boolean m_unpruned = false;

  // Confidence level
  private float m_CF = 0.25f;
```

```
// Minimum number of instances
private int m_minNumObj = 2;

// Determines whether probabilities are smoothed using
// Laplace correction when predictions are generated
private boolean m_useLaplace = false;

// Use reduced error pruning?
private boolean m_reducedErrorPruning = false;

// Number of folds for reduced error pruning.
private int m_numFolds = 3;

// Binary splits on nominal attributes?
private boolean m_binarySplits = false;

// Subtree raising to be performed?
private boolean m_subtreeRaising = true;

// Cleanup after the tree has been built.
private boolean m_noCleanup = false;

// Random number seed for reduced-error pruning.
private int m_Seed = 1;

// Returns a string describing classifier

public String globalInfo() {

  return  "Class for generating a pruned or unpruned C4.5 decision tree. For
more "
    + "information, see\n\n"
    + "Ross Quinlan (1993). \"C4.5: Programs for Machine Learning\", "
    + "Morgan Kaufmann Publishers, San Mateo, CA.\n\n";
}

public void buildClassifier(Instances instances)
     throws Exception {
      //System.out.println("jkfgkjfdgdfnfgjk;");

  ModelSelection modSelection;
modSelection = new BinC45ModelSelection(m_minNumObj, instances);
```

```
  m_root = new PruneableClassifierTree(modSelection, !m_unpruned,
m_numFolds,
                                       !m_noCleanup, m_Seed);



  m_root.buildClassifier(instances);
 }

 // Classifies an instance.

 public double classifyInstance(Instance instance) throws Exception {

  return m_root.classifyInstance(instance);
 }

 // Returns class probabilities for an instance.

 public final double [] distributionForInstance(Instance instance)
    throws Exception {

  return m_root.distributionForInstance(instance, m_useLaplace);
 }

 // Returns graph describing the tree.

 public String graph() throws Exception {

  return m_root.graph();
 }

 // Returns tree in prefix order.

 public String prefix() throws Exception {

  return m_root.prefix();
 }


 // Returns tree as an if-then statement.

 public String toSource(String className) throws Exception {
```

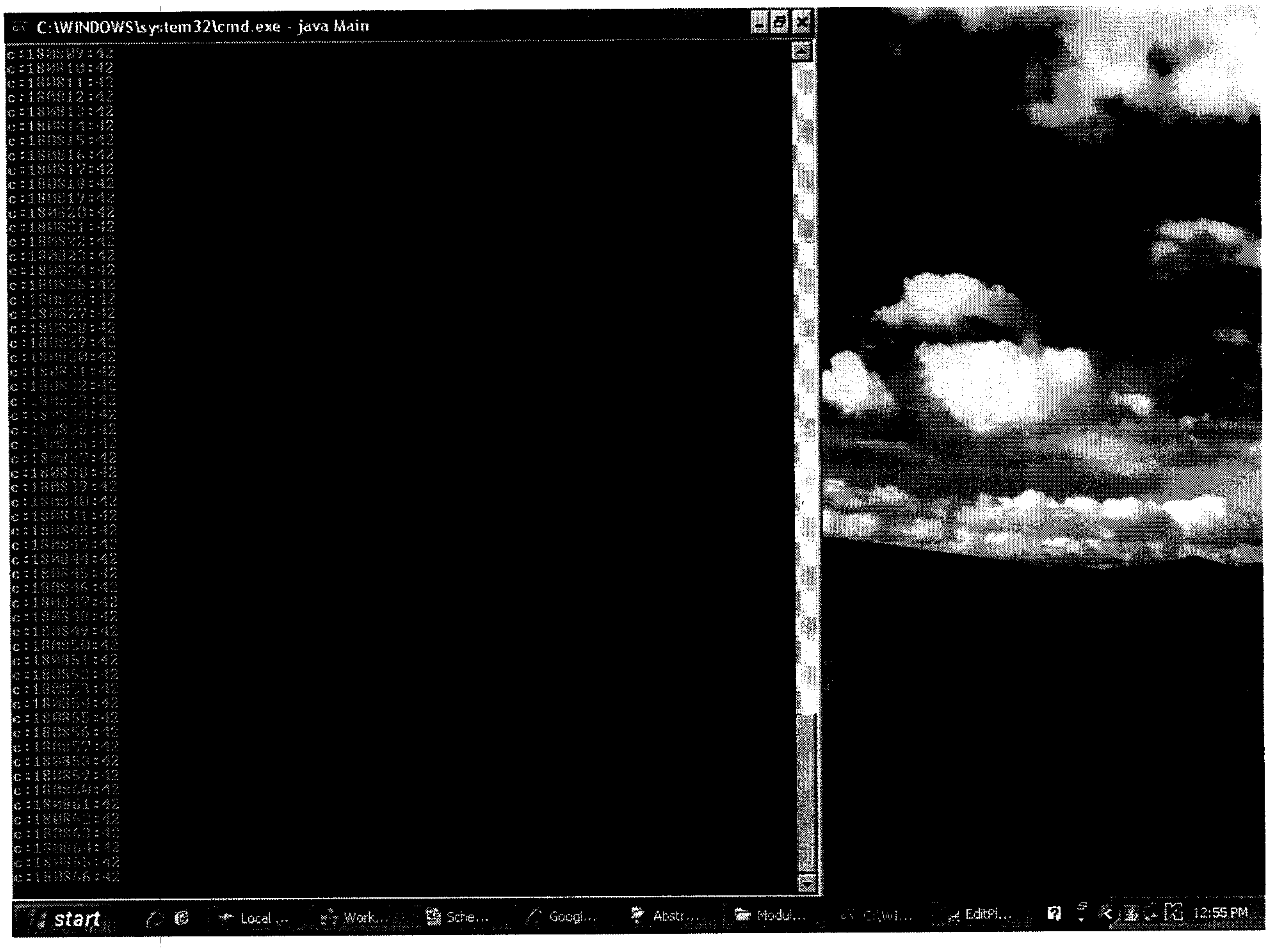
```
 StringBuffer [] source = m_root.toSource(className);
 return
 "class " + className + " {\n\n"
 +"  public static double classify(Object [] i)\n"
 +"    throws Exception {\n\n"
 +"    double p = Double.NaN;\n"
 + source[0]  // Assignment code
 +"    return p;\n"
 +"  }\n"
 + source[1]  // Support code
 +"}\n";
}

// Parses a given list of options.

public void setOptions(String[] options) throws Exception {

  String minNumString = Utils.getOption('M', options);
  if (minNumString.length() != 0) {
   m_minNumObj = Integer.parseInt(minNumString);
  } else {
   m_minNumObj = 2;
  }
  m_binarySplits = Utils.getFlag('B', options);
  m_useLaplace = Utils.getFlag('A', options);

  // Pruning options
  m_unpruned = Utils.getFlag('U', options);
  m_subtreeRaising = !Utils.getFlag('S', options);
  m_noCleanup = Utils.getFlag('L', options);
  if ((m_unpruned) && (!m_subtreeRaising)) {
    throw new Exception("Subtree raising doesn't need to be unset for
unpruned tree!");
  }
  m_reducedErrorPruning = Utils.getFlag('R', options);
  if ((m_unpruned) && (m_reducedErrorPruning)) {
    throw new Exception("Unpruned tree and reduced error pruning can't be
selected " +
                        "simultaneously!");
  }
  String confidenceString = Utils.getOption('C', options);
  if (confidenceString.length() != 0) {
    if (m_reducedErrorPruning) {
      throw new Exception("Setting the confidence doesn't make sense " +
                          "for reduced error pruning.");
    } else if (m_unpruned) {
      throw new Exception("Doesn't make sense to change confidence for
unpruned "
                          +"tree!");
    } else {
      m_CF = (new Float(confidenceString)).floatValue();
      if ((m_CF <= 0) || (m_CF >= 1)) {
        throw new Exception("Confidence has to be greater than zero and
smaller " +
                            "than one!");
      }
    }
  } else {
    m_CF = 0.25f;
  }
  String numFoldsString = Utils.getOption('N', options);
  if (numFoldsString.length() != 0) {
    if (!m_reducedErrorPruning) {
      throw new Exception("Setting the number of folds" +
                          " doesn't make sense if" +
                          " reduced error pruning is not selected.");
    } else {
      m_numFolds = Integer.parseInt(numFoldsString);
    }
  } else {
    m_numFolds = 3;
  }
  String seedString = Utils.getOption('Q', options);
  if (seedString.length() != 0) {
    m_Seed = Integer.parseInt(seedString);
  } else {
    m_Seed = 1;
  }
}

// Gets the current settings of the Classifier.

public String [] getOptions() {

  String [] options = new String [14];
  int current = 0;

  if (m_noCleanup) {
   options[current++] = "-L";
  }
  if (m_unpruned) {
   options[current++] = "-U";
  } else {
   if (!m_subtreeRaising) {
      options[current++] = "-S";
   }
   if (m_reducedErrorPruning) {
      options[current++] = "-R";
      options[current++] = "-N"; options[current++] = "" + m_numFolds;
      options[current++] = "-Q"; options[current++] = "" + m_Seed;
   } else {
      options[current++] = "-C"; options[current++] = "" + m_CF;
   }
  }
  if (m_binarySplits) {
   options[current++] = "-B";
  }
  options[current++] = "-M"; options[current++] = "" + m_minNumObj;
  if (m_useLaplace) {
   options[current++] = "-A";
  }

  while (current < options.length) {
   options[current++] = "";
  }
  return options;
 }

 public String seedTipText() {
  return "The seed used for randomizing the data " +
   "when reduced-error pruning is used.";
 }

 public int getSeed() {

  return m_Seed;
 }
```

```
// Set the value of Seed.

public void setSeed(int newSeed) {

  m_Seed = newSeed;
}

public String useLaplaceTipText() {
  return "Whether counts at leaves are smoothed based on Laplace.";
}

// Get the value of useLaplace.

public boolean getUseLaplace() {

  return m_useLaplace;
}

// Set the value of useLaplace.

public void setUseLaplace(boolean newuseLaplace) {

  m_useLaplace = newuseLaplace;
}

// Returns a description of the classifier.

public String toString() {

  if (m_root == null) {
    return "No classifier built";
  }
  if (m_unpruned)
    return "C45 unpruned tree\n------------------\n" + m_root.toString();
  else
    return "C45 pruned tree\n------------------\nRule\n" + m_root.toString();
}

// Returns a superconcise version of the model

   public String toSummaryString() {

  return "Number of leaves: " + m_root.numLeaves() + "\n"
```

```
    + "Size of the tree: " + m_root.numNodes() + "\n";
}

public double measureTreeSize() {
  return m_root.numNodes();
}

// Returns the number of leaves

public double measureNumLeaves() {
  return m_root.numLeaves();
}

// Returns the number of rules (same as number of leaves)

public double measureNumRules() {
  return m_root.numLeaves();
}

// Returns an enumeration of the additional measure names

public Enumeration enumerateMeasures() {
  Vector newVector = new Vector(3);
  newVector.addElement("measureTreeSize");
  newVector.addElement("measureNumLeaves");
  newVector.addElement("measureNumRules");
  return newVector.elements();
}

// Returns the value of the named measure

public double getMeasure(String additionalMeasureName) {
  if
(additionalMeasureName.compareToIgnoreCase("measureNumRules")==0)
{
    return measureNumRules();
  } else if
(additionalMeasureName.compareToIgnoreCase("measureTreeSize") == 0)
{
    return measureTreeSize();
  } else if
(additionalMeasureName.compareToIgnoreCase("measureNumLeaves") ==
0) {
    return measureNumLeaves();
  } else {
    throw new IllegalArgumentException(additionalMeasureName
                          + " not supported (c45)");
  }
}

}

// Get the value of unpruned.

public boolean getUnpruned() {

  return m_unpruned;
}

public void setUnpruned(boolean v) {

  if (v) {
    m_reducedErrorPruning = false;
  }
  m_unpruned = v;
}

  public void setConfidenceFactor(float v) {

  m_CF = v;
}

public String minNumObjTipText() {
  return "The minimum number of instances per leaf.";
}

// Get the value of minNumObj.

public int getMinNumObj() {

  return m_minNumObj;
}

// Set the value of minNumObj.

public void setMinNumObj(int v) {
  m_minNumObj = v;
}


public String reducedErrorPruningTipText() {
  return "Whether reduced-error pruning is used instead of C.4.5 pruning.";
}

// Get the value of reducedErrorPruning.

public boolean getReducedErrorPruning() {

  return m_reducedErrorPruning;
}

public void setReducedErrorPruning(boolean v) {

  if (v) {
    m_unpruned = false;
  }
  m_reducedErrorPruning = v;
}

public String numFoldsTipText() {
  return "Determines the amount of data used for reduced-error pruning. "
    + " One fold is used for pruning, the rest for growing the tree.";
}

// Get the value of numFolds.

public int getNumFolds() {

  return m_numFolds;
}

// Set the value of numFolds.

public void setNumFolds(int v) {
  m_numFolds = v;
}

public String binarySplitsTipText() {
  return "Whether to use binary splits on nominal attributes when "
    + "building the trees.";
}

// Get the value of binarySplits.

public boolean getBinarySplits() {

  return m_binarySplits;
}

// Set the value of binarySplits.

public void setBinarySplits(boolean v) {

  m_binarySplits = v;
}

public String subtreeRaisingTipText() {
  return "Whether to consider the subtree raising operation when pruning.";
}

// Get the value of subtreeRaising.

public boolean getSubtreeRaising() {

  return m_subtreeRaising;
}
public void setSubtreeRaising(boolean v) {

  m_subtreeRaising = v;
} }
public boolean getSaveInstanceData() {

  return m_noCleanup;
}
 public void setSaveInstanceData(boolean v) {

  m_noCleanup = v;
}
 public Enumeration listOptions() {
    throw new UnsupportedOperationException("Not supported yet.");
  }}
```

7.2 SCREENSHOTS:

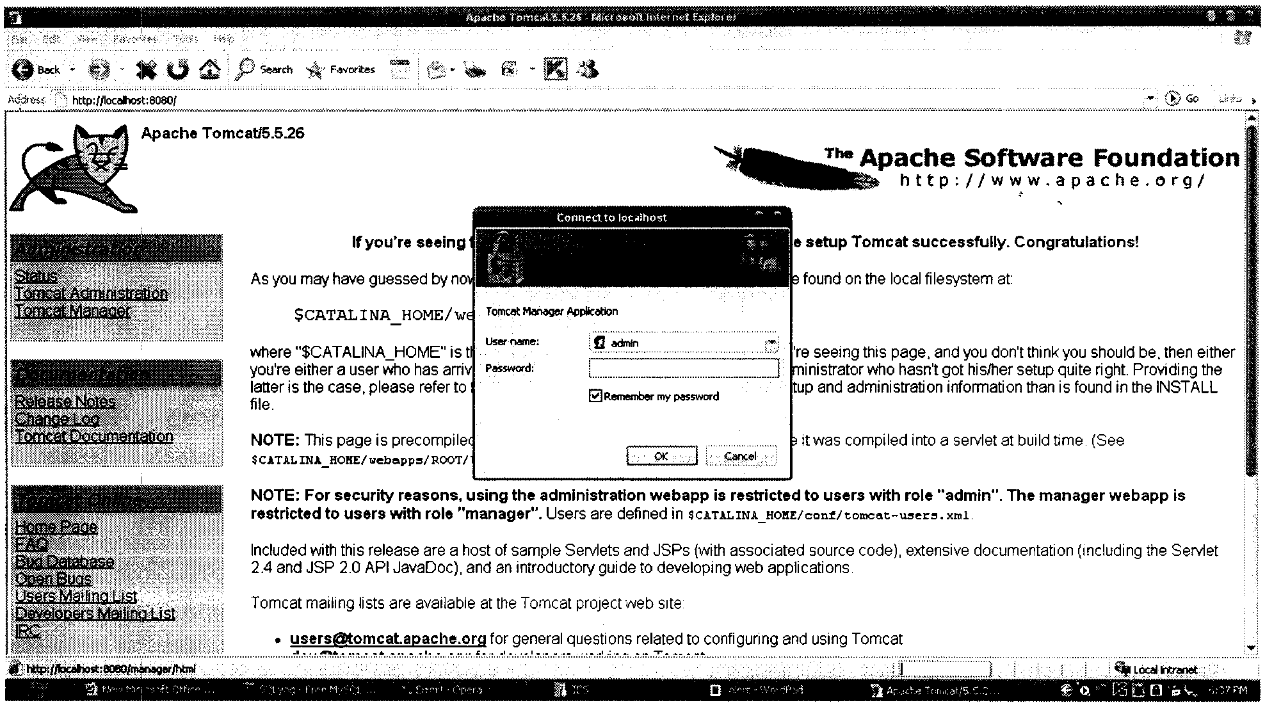
Packet capture :



```
c:188809:42
c:188810:42
c:188811:42
c:188812:42
c:188813:42
c:188814:42
c:188815:42
c:188816:42
c:188817:42
c:188818:42
c:188819:42
c:188820:42
c:188821:42
c:188822:42
c:188823:42
c:188824:42
c:188825:42
c:188826:42
c:188827:42
c:188828:42
c:188829:42
c:188830:42
c:188831:42
c:188832:42
c:188833:42
c:188834:42
c:188835:42
c:188836:42
c:188837:42
c:188838:42
c:188839:42
c:188840:42
c:188841:42
c:188842:42
c:188843:42
c:188844:42
c:188845:42
c:188846:42
c:188847:42
c:188848:42
c:188849:42
c:188850:42
c:188851:42
c:188852:42
c:188853:42
c:188854:42
c:188855:42
c:188856:42
c:188857:42
c:188858:42
c:188859:42
c:188860:42
c:188861:42
c:188862:42
c:188863:42
c:188864:42
c:188865:42
c:188866:42
```

Fig 6 : Packet capturing

Apache Verification :


Apache Tomcat/5.5.26 - Microsoft Internet Explorer

Address http://localhost:8080/

Apache Tomcat/5.5.26

The Apache Software Foundation
http://www.apache.org/

Administration
Status
Tomcat Administration
Tomcat Manager

Documentation
Release Notes
Change Log
Tomcat Documentation

Tomcat Online
Home Page
FAQ
Bug Database
Open Bugs
Users Mailing List
Developers Mailing List
IRC

If you're seeing **e setup Tomcat successfully. Congratulations!**

As you may have guessed by no e found on the local filesystem at:

```
$CATALINA_HOME/we
```

where "$CATALINA_HOME" is t 're seeing this page, and you don't think you should be, then either you're either a user who has arriv ministrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to tup and administration information than is found in the INSTALL file.

NOTE: This page is precompile e it was compiled into a servlet at build time. (See `$CATALINA_HOME/webapps/ROOT/`

NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation (including the Servlet 2.4 and JSP 2.0 API JavaDoc), and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- **users@tomcat.apache.org** for general questions related to configuring and using Tomcat

Connect to localhost

Tomcat Manager Application

User name: admin

Password:

[✓] Remember my password

OK Cancel

http://localhost:8080/manager/html

Local intranet

Fig 7 : apache verification

Login :

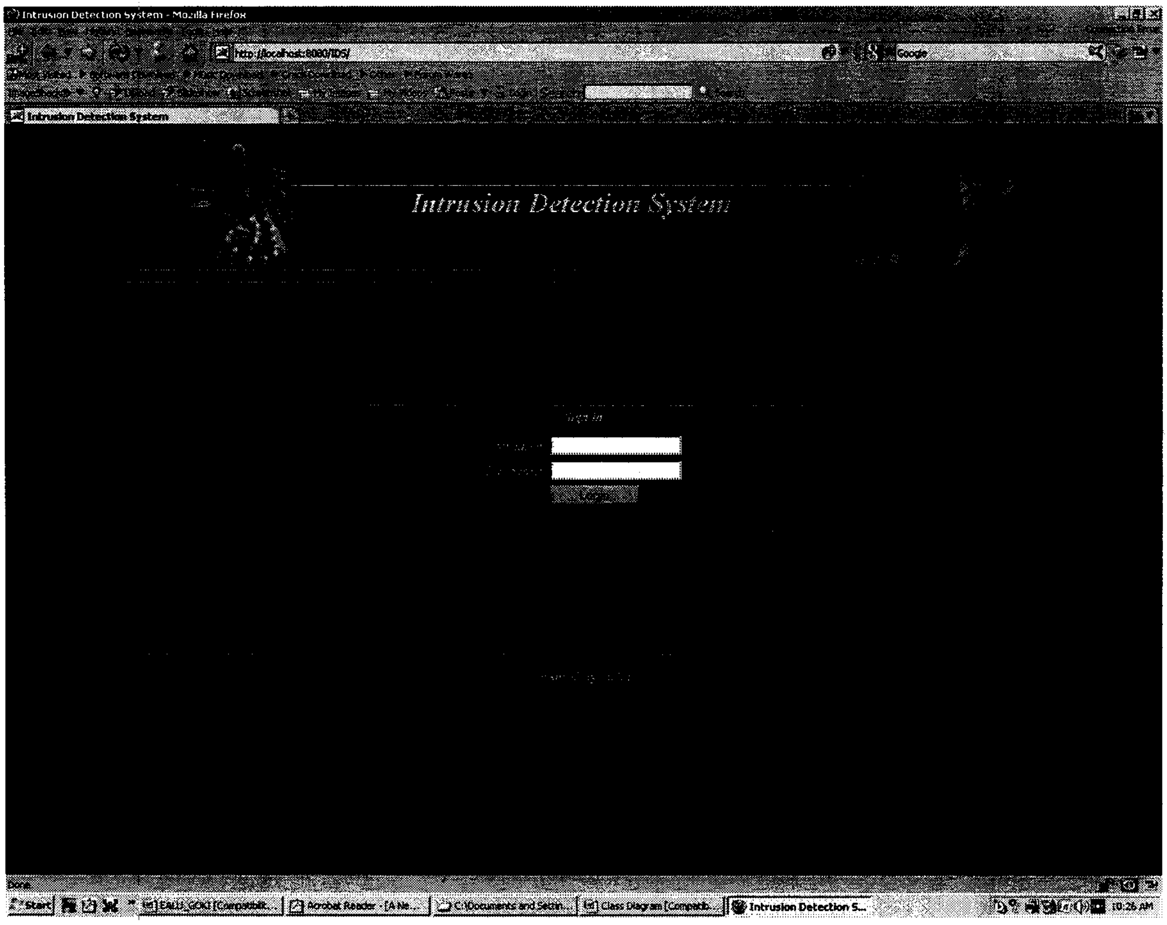
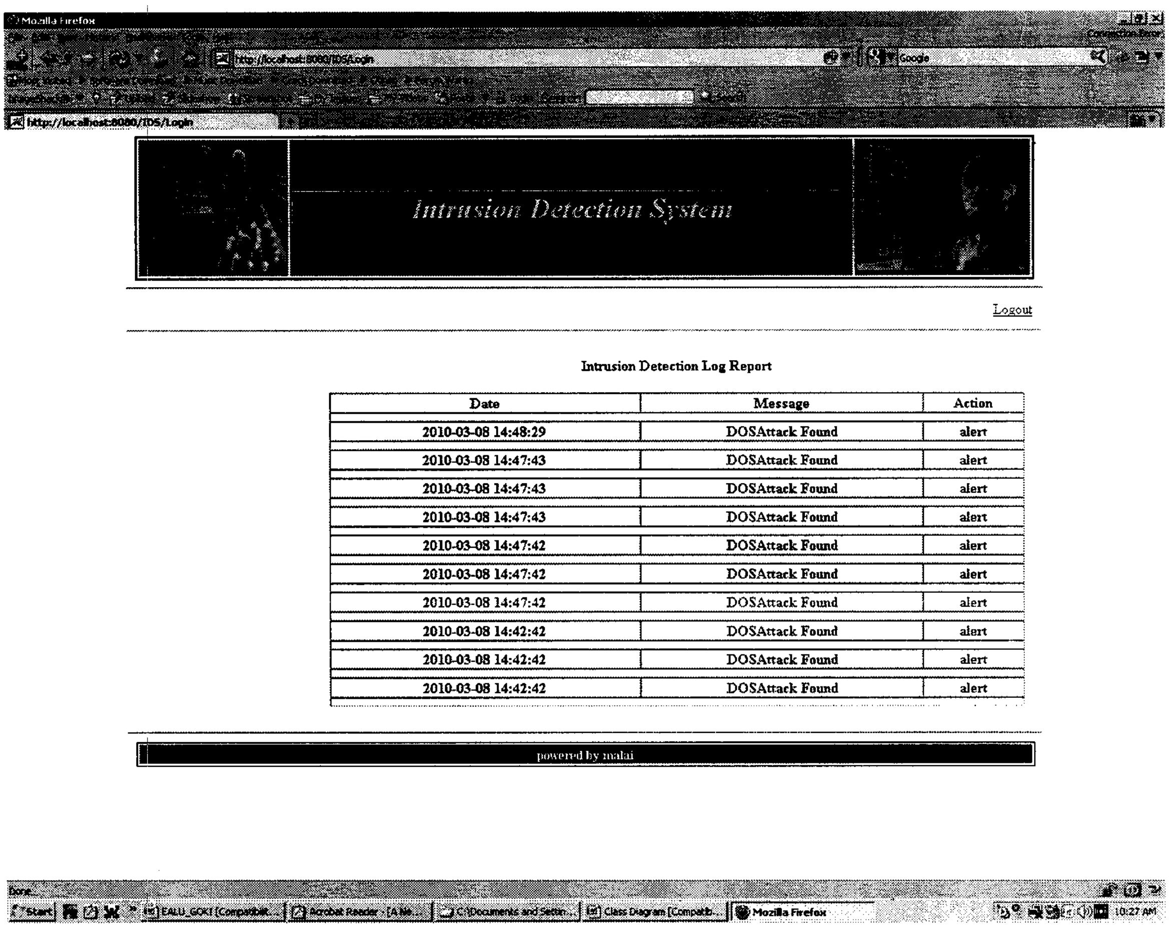

Intrusion Detection System - Mozilla Firefox
http://localhost:8080/IDS/
Intrusion Detection System
Intrusion Detection System
Login

Fig.8 : Login page

Report 1:

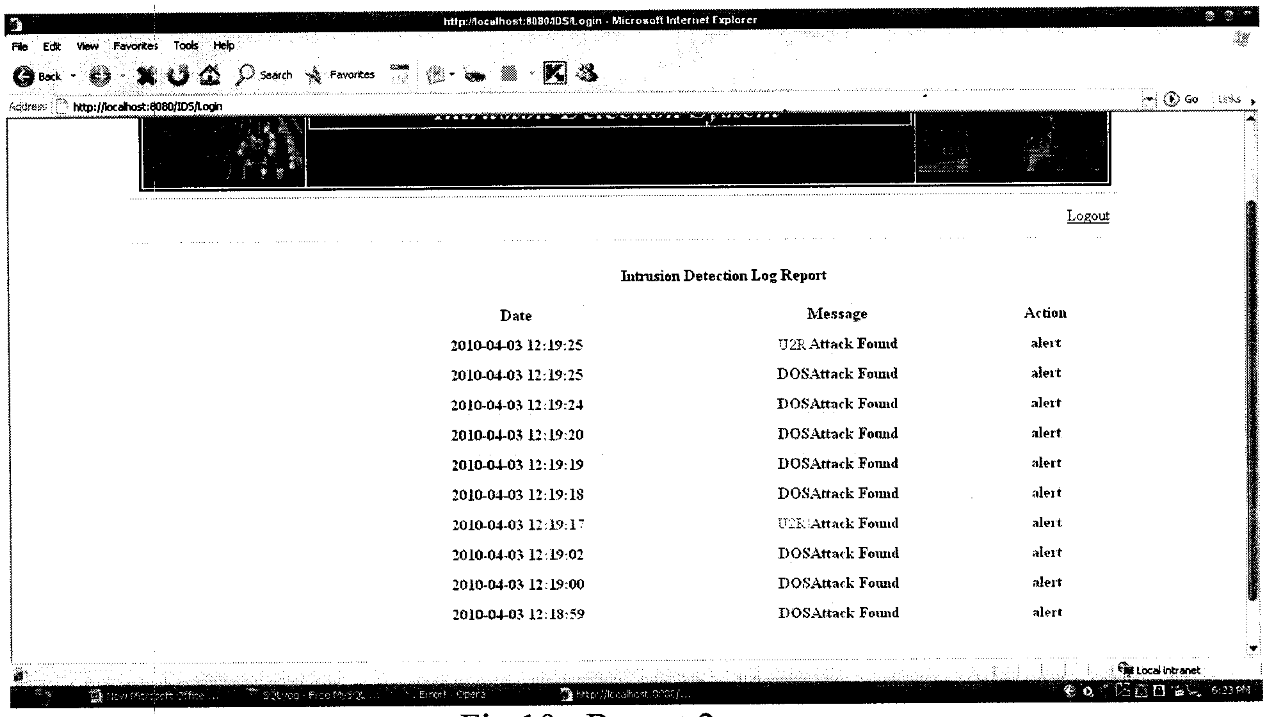


Intrusion Detection Log Report

Date	Message	Action
2010-03-08 14:48:29	DOSAttack Found	alert
2010-03-08 14:47:43	DOSAttack Found	alert
2010-03-08 14:47:43	DOSAttack Found	alert
2010-03-08 14:47:43	DOSAttack Found	alert
2010-03-08 14:47:42	DOSAttack Found	alert
2010-03-08 14:47:42	DOSAttack Found	alert
2010-03-08 14:47:42	DOSAttack Found	alert
2010-03-08 14:42:42	DOSAttack Found	alert
2010-03-08 14:42:42	DOSAttack Found	alert
2010-03-08 14:42:42	DOSAttack Found	alert

Fig.9 : Report

Report 2:


http://localhost:8080/IDS/Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/IDS/Login
Go Links
Logout
Local intranet

Intrusion Detection Log Report

Date	Message	Action
2010-04-03 12:19:25	U2R Attack Found	alert
2010-04-03 12:19:25	DOSAttack Found	alert
2010-04-03 12:19:24	DOSAttack Found	alert
2010-04-03 12:19:20	DOSAttack Found	alert
2010-04-03 12:19:19	DOSAttack Found	alert
2010-04-03 12:19:18	DOSAttack Found	alert
2010-04-03 12:19:17	U2R Attack Found	alert
2010-04-03 12:19:02	DOSAttack Found	alert
2010-04-03 12:19:00	DOSAttack Found	alert
2010-04-03 12:18:59	DOSAttack Found	alert

Fig 10 : Report 2

REFERENCES

- W. Lee, S. J. Stolfo, and K. W. Mok "A data mining framework for building intrusion detection models", Proc. of the 1999 IEEE Symp. on Security and Privacy, Oakland, CA, May, 1999, pp. 120-132.

- W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection", Proc. of the 7th USENIX Security Symp., San Antonio, TX, 1998.

- S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems", Computer & Security, Vol. 24, Issue 4, June 2005, pp. 295-307.

- J. R. Quinlan, "C4.5: programs for machine learning", Morgan Kaufmann Publishers, 1993.

- G. I. Webb, "Multiboosting: a technique for combining boosting and wagging", Machine Learning, Vol. 40, 2000, pp. 159-196.

Web References :

- www.developer.com
- www.javaworld.com
- www.en.wikipedia.org