Developing a Host Intrusion Prevention System by Using Data Mining

تطوير مضيّف لنظام منع التطفل باستخدام تنقيب البيانات

By

Tahani Nawaf Alawneh

Supervisor

Prof. Dr. Alaa Hussein Al-Hamami

Submitted in partial fulfillment of the requirements for the degree of Master in computer science

Graduate Collage of Computing Studies

Amman Arab University for Graduate Studies

Febreuary-2010

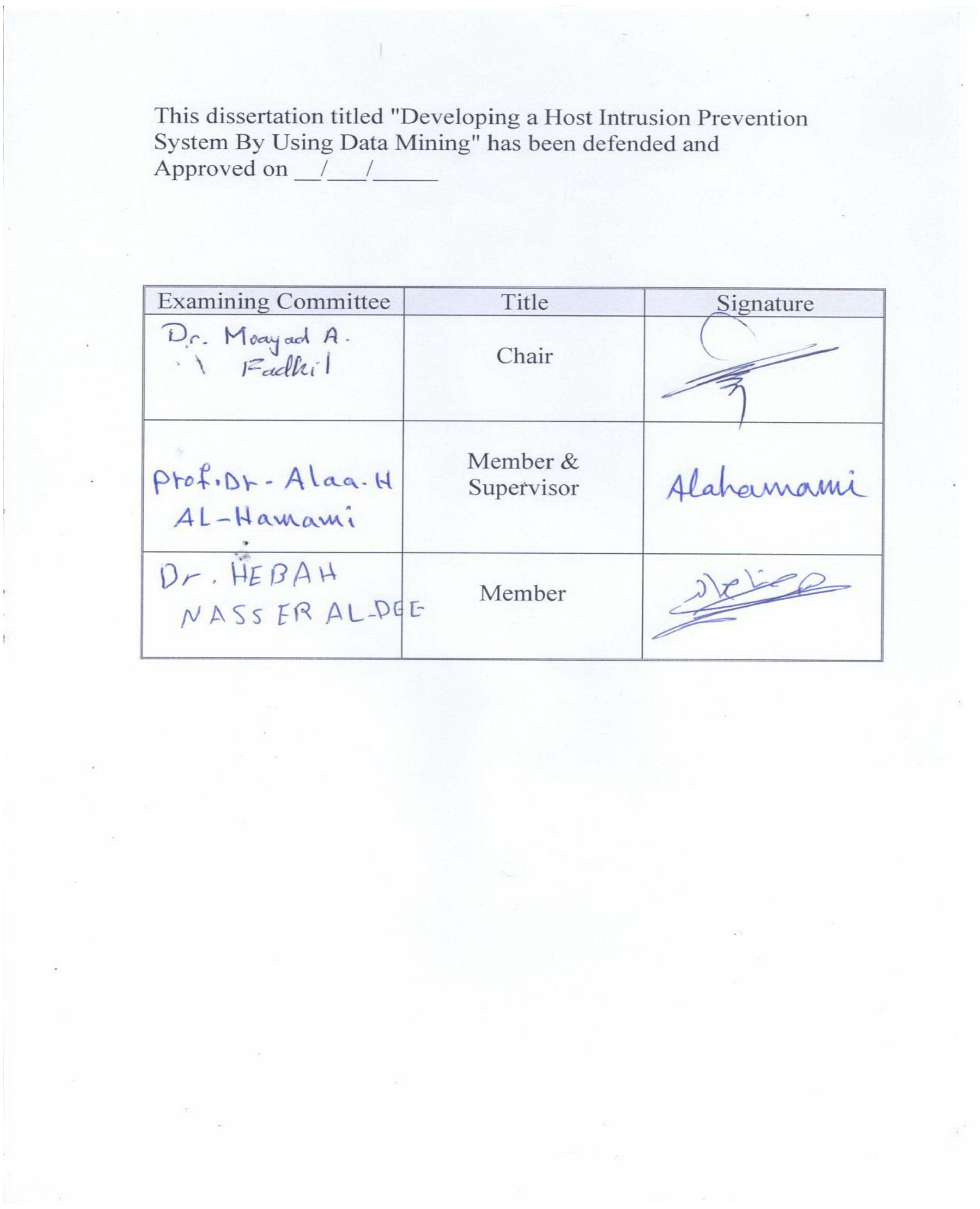
AUTHORIZATION

I, Tahani Alawneh the undersigned, herby authorize Amman Arab University For Graduate Studies to provide libraries, organizations, institutions and individuals with copies of my thesis when requested.

Tahani N. Alawneh

[illegible]

24/4/2010



This dissertation titled "Developing a Host Intrusion Prevention System By Using Data Mining" has been defended and Approved on __/___/_____

Examining Committee	Title	Signature
Dr. Moayad A. Fadhil	Chair	[illegible]
Prof.Dr- Alaa.H AL-Hamami	Member & Supervisor	Alahamami
Dr. HEBAH NASSER AL-DEE	Member	[illegible]

Dedication

اللهم أوزعني أن أشكر نعمتك التي أنعمت علي وعلى والدي وأن أعمل صالحاً ترضاه

To my parents, whom I will be grateful to till the end of my life.

To my beloved brothers and sisters and their families.

To my Compassionate husband for his support and patience,

To the two pure followers in my life Abed-Alrahman and Mareia**.**

Tahani

Acknowledgment

Hope these few words could carry my true thanks to all who helped me finish this dissertation.

The first one I would like to thank from my heart is Prof. Alaa Alhamai for his trust, kindness, guidance, and patience, for being not just a wise advisor, but also a kind father.

I would like to thank my best friend, Rawan AlKhatib, without her help, I would not have been here today. Thanks for all her support and wisdom.

Kaleel Hamid, thank you my friend for your brilliant ideas.

I am very grateful to my colleagues in the IT department in **ASEZA** (Aqaba Special Economic Zone Authority) for supporting me with this study , and for their help and trust.

Table of Contents

List of Abbreviations

ARFF	Attribute Relation File Format
CLI	Command Line Interface
CLI	Command Line Interface
COM	Component Object Model
COM	Component Object Model
DAID	(Database-centric Architecture for Intrusion Detection)
GUI	Graphical User Interface
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IT	Information Technology
PDA	Personal Digital Assistance
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VPN	Virtual Private Network
XML	Extensible Markup Language

List of Figures

List of Tables

ABSTRACT

Intrusion Prevention System (**IPS**), as a security solution have their own unique characteristics in analyzing, detecting and preventing intruders' acts, which provide a quite good service in securing the network. These special characteristics made **IPS** a new trend for researches at the moment.

Host based **IPS** mostly depend on a static signature mechanism to identify intruders, which in turn needs to be updated from time to time to insure the most accurate detection.

This study aims to enhance intruders' detection, by replacing the static database with a dynamic one, and even more adding inelegance to the detecting mechanism through data mining. A feedback to the whole process is being made to help in making future inspections to be more realistic.

All the above will contribute to achieving enhancement in the following:

- Evolving the techniques of investigating activities due to the using of data mining.
- Integrate or could eliminate antivirus programs installed on **PCs**.
- Maximize the level of security of the whole network, through securing single hosts.

Arabic Summary

الملخـص

يتميز نظام منع التطفل بخصائص فريدة من حيث تحليل الأخطار و تحديدها و منعها، مما جعله وسيلةً جيدةً يعتمد عليها لتحقيق الحماية لشبكات الحاسوب، و هذه الخصائص هي ذاتها التي جعلت نظام منع التطفل توجهاً جديداً للباحثين.

إن نظام منع التطفل المطبق على مستوى المضيف يعتمد في عمله من حيث آلية التعرف على الأخطار على قاعدة بيانات ثابتة و التي تحتاج إلى تحديث من و قت لآخر.

إن هذه الدراسة تهدف إلى تحسين آلية التعرف على الأخطار من خلال استبدال قاعدة البيانات الثابتة بأخرى ديناميكية، إضافةً إلى إدخال نوع من الذكاء إلى هذه العملية عن طريق تنقيب البيانات. و لجعل عملية التعرف على الخطار أقرب إلى الواقع بأكبر ما يمكن فقد تم إضافة تغذية راجعة لقاعدة البيانات السابقة. و بناءً على ما سبق ذكره فإن البحث سيسهم بما يلي:

تطوير آلية التعرف على الأخطار بفضل استخدام تنقيب البيانات.
التكامل مع برامج الحماية من الفايوسات أو حتى استبدالها كلياً.
زيادة مستوى أمن الشبكة عن طريق تأمين كافة الحواسيب المربوطة على الشبكة.

Chapter One
Introduction

1.1- Introduction

Information systems are more powerful today than ever before. People increasingly rely on Information Technology (**IT**) that spreads exponentially every day, for example, using internet, one could shop online, or pay his taxes, or even read the news paper. Since the internet, being prevalent, is the target, there are, still some consequences indeed, as the value of related information resources grows up, the risks related grows up in parallel tremendously, which makes information resources protection a very critical matter, and so providing the needed security for end users is essential today.

Fifty years ago the term security in **IT**, was all about physical practices, one could secure his data center room that has huge main frames by securing its access using a guard for example. But when it became necessary to connect private networks together with all evolutions related to end up with internet today, security is no longer a physical term. To reach data, intruders should not pass a physical entrance, or even plug in a network cable inside your own place with the existence of wireless technology instead.

Security risks come in the forms of Viruses, Trojan Horses, Malwares, Application security holes, or from a Hacker or a Cracker who tries to mess up with your data. 80 percent of security breaks comes from internal intruders [1]. Since 1995, the annual increase in risk from internet hacking is up 60% per year, while the annual increase in risk from viruses and worms is up over 100% per year [10].

Accordingly, there are many solutions to stand against such intruders, such as antivirus, firewall, Spyware, security patches, Intrusion Detection Systems (**IDS**) and intrusion Prevention Systems (**IPS**).

Although both Intrusion Prevention Systems (**IPS)** and Intrusion Detection Systems (**IDS)** can detect an intruder, IPS have a plus for preventing intrusion by taking certain actions once it is being detected. Among all security solutions, **IPS** have their own special characteristics in analyzing, detecting and preventing intruders' acts, which provide a quite good service in securing the network. These special characteristics made **IPS** a new trend for researches at the moment, many papers have discussed ways for choosing the best **IPS**, and other papers discussed new attack detecting mechanisms.

Another concept that is being growing so fast in **IT** fields, is data mining, data mining is the analysis of (often large) observational data sets to find
unsuspected relationships, and to summarize the data in novel ways that are both understandable and useful to the data owner [7], data mining is being considered as a revolutionary field in developing different **IT** solutions today.

This study aims to enhance intruders' detection, by using data mining approaches, namely decision tree. The input for the decision tree algorithm is a comma separator file format that has entries for the most infected or targeted resources by intruders, whether an intruder was a Virus (a type of malicious software that can destroy the computer's hard drive, files, and programs in memory, and that replicates itself to other disks), a Trojan horse (computer program that is apparently or actually useful and contains a harmful code), a Spayware (a program that collects data about your PC or network and sends it to another destination), or even Human acts. This file was prepopulated with entries that indicate the most famous intrusion infection symptoms. But it keeps growing each time an anomaly in one of these resources is being detected. With decision tree, the file is being treated to decide whether the resource behavior was due to a normal action or an intruder action. The user is informed, and chooses to commit the decision or not, at that time the file is being updated depending on the user response, which provides a feedback to the whole process and definitely causes future inspections to be more realistic.

WEKA workbench was used to perform data mining processing. WEKA stands for Waikato Environment for Knowledge Analysis, where it was developed at the University of Waikato in New Zealand. WEKA is a collection of machine learning algorithms for data mining; it is an open source machine learning software written in Java, It contains tools for a whole range of data mining tasks like data pre-processing, Classification and Visualization

WEKA provides all the processes of data mining techniques, starting from preparing the data, visualizing both inputs and outputs, and evaluating the learning algorithms.

The WEKA implementation of the used decision tree algorithm is **J48**. Decision tree is one of data mining operations that is applied to categorical attributes, it aims to predict the value of one attribute by means of some of the other available attributes (categorical attributes: attributes that takes two or more discrete values).

System alerts tool in windows was used as a trigger to fill the input data file whenever an anomaly behavior in certain system resources is being noticed. This tool is one among many powerful tools available in the control panel under the Administrative tools.

1.2-Intrusion Prevention Overview

Intrusion Prevention Systems (**IPS**) are security protection devices or applications that can prevent attacks against your network devices [9]. The first show of **IPS** was as an added feature to other security products such as antiviruses and firewalls, but it continued evolving until it became an independent solution by itself; where you can find two types: host and network.

Through the history of technology, computer systems kept integrating with all kinds of human activities, this ended up with powerful servers storing and handling all kinds of critical information ever needed.

Valuable things are always threatened by theft, and so is information, and these threats kept growing and growing as information systems became more available to users, in old days mainframes had only a limited access by certain terminals, which limited the access to its information only to the physical access to those terminals, so security was a very easy thing to keep if only that access was secured physically, but as time went on, and as **IT** industry kept evolving, computer and related devices became cheaper, and internet became available and quite cheap too, now security is no longer a matter of preventing physical access to resources which can simply be accomplished by hiring a guard, but is a much more complex problem which needs a series of procedures and activities, in addition to dedicated solutions and products that insure the protection against any attacks [11].

Many companies use firewalls to limit the external access to their network, a firewall is a network device that shields the trusted network from unauthorized users in the un-trusted network by blocking certain specific types of traffic [18]. So a firewall either software or hardware, acts as a gate guard, it does not investigate what happens inside the network after a successful access, which is not enough especially with the existence of wireless network that does not stop by the end of your physical walls, where intruders can gain access directly to your resources from down street if it is not secured properly

1.2.1- IPS importance

When trying to understand why **IPS** are necessary in todays' networks, you need to consider the following factors [9]:

1. Technology adoption.
2. Target value.
3. Attacking techniques.

1. Technology adoption**:** client-server, Internet, wireless and mobile computing, have affected security level severely, which in turn affected **IPS** needs and technologies.

a. Client-server: Because it has a distributed architecture with a non-centralized control, it opened new ways for attacking which did not exist when dump terminals were used.

b. Internet: Connecting all computers and servers globally is a big risk indeed, add to that, the most used protocols are Transmission Control Protocol (**TCP**) and User Datagram Protocol (**UDP**) which both send data through the network unencrypted, while **UDP** is connectionless, which gives an easy chance to spoof the source address.

c. Wireless technology: There is no need any more to plug a cable physically into a switch port to be able to use a network resources, with wireless technology you can easily connect to the network if you receive the access point signals, even if you are an attacker.

d. Mobile Computing: When enabling your user to keep working from outside the office as he is really there, it is a great facility, but with a great risk too. Laptops, mobile phones, Personal Digital Assistance (**PDA**), can connect to the network using Virtual Private Network (**VPN**), and firewalls, but it is still considered as unsafe access, because no more monitoring is done for authorized login.

All of these reasons made the existence of **IPS** very essential in the network, to investigate farther more about internal traffic.

2. Target Value: Theft in **IT** field is not because of the hardware value, it is because of the stored information indeed. Computers are used in all facilities nowadays, government, banks, and other private sectors. With the competition of providing the best services ever, intruders can find there ways to access valuable data.

3. Attacking techniques: Not only the computer technology evolves, but also attacking techniques are evolving too, the way attackers access the network, and the complexity of discovering it added to that its impact made it very important to insure a reliable solution that can guarantee an accepted level of protection.

Attackers used different methods to reach their goals, they moved from physical access, to using the internet, media, and emails; they even developed specific attacks such as **SQL** database servers. The attacking mitigation tools did not evolve enough to keep track, so **IPS** are developed to follow the attackers.

1.2.2- IPS differences

As mentioned before **IPS** have two types: Network and Host. Network **IPS** investigate network traffic, while Host **IPS** investigate individual computers activity. But that makes Host **IPS** different from many solutions used for securing hosts such as: antivirus, firewalls and Intrusion Detection Systems (**IDS**).

1. Antivirus works by examining files against a defined database, that contains all virus patterns, although it is the most used method to protect host, but it still suffers from some shortage in its mechanism, antivirus vendors should always update their databases against any new attacks, which means in turn that for any new virus they should analyze it and update the database, so users will be on the safe side. But it is not always guaranteed to find a virus and update the database without victims.

2. Firewalls work same as antivirus except they check incoming network traffic against a defined database, plus they have the ability to block unauthorized connection, which can protect the network from being infected by any host. But firewalls still have the same problem of updating their database, and for connection blocking feature there are some attacks that uses legal connection to access the network, such as email for example.

3. IDS works as a good partner with firewall, as it keeps investigating any suspicious activity, even after an authorized connection. But on the other hand if any damage happens to any system, then it must be fixed, which makes the process of preventing the damage before occurring much easier than fixing it.

4. IPS covers all the shortage in the above mentioned solutions, which gives great flexibility when dealing with host security threats.

A Host Based Intrusion Prevention System (**HIPS**) has the following Capabilities [9]:

1. Blocks malicious code actions

HIPS does not only generate alarms when a malicious code is detected but, more important, is able to stop the attack and so cut its life cycle.

2. Doesn't disrupt normal operations

HIPS must not change the normal operation and functionality of processes in order to keep security around, indeed it must be able to detect unnormal operations and block them rather than blocking the whole process, for example, **HIPS** should not prevent all attachments in email messages, but it should be able to detect malicious ones and block them.

3. Distinguishes between attacks and normal events

HIPS should be able to find out if an action is an attack or a normal operation, there might be some positive restrictions which cause a higher degree of security when first implementing **HIPS**, which can be altered later on, but it is not expected to find out a negative restriction, that causes a threat to attack a device.

1.2.3- Signature

One feature that is always related to **IPS** is signature. **IPS** differs in three ways when dealing with signatures according to signature type, signature trigger and signature action.

A. Signature Types

There are two signature types for **IPS**: atomic signature and stateful signature. The only difference between both is in the inspection process, if **IPS** needs to keep history or previous state, then it would be stateful, if not, then it will be an atomic.

As an example for host-based **IPS** for both types, consider the following: a common way to inspect any anomalous user behavior, is to establish a baseline for the operations that the user usually performs. Then **IPS** monitors any deviation about the base line, so if **IPS** triggered a signature action when let say the user invokes a system call that is not usually called, this **IPS** is an example for host-based atomic signatures. But if **IPS** triggered a signature when a command shell is being called after this system call then it is a host-based stateful signature

B. Signature Triggers

Triggering mechanisms are the most important thing in **IPS**, and these are the conditions that cause **IPS** to trigger a signature action. They are as follows:

- Pattern detection.
- Anomaly-based detection.
- Behavior-based detection.

(1) Pattern detection**:** It is the simplest mechanism of all, where it identifies a specified pattern. This pattern might be a string, or it can be a sequence of system calls.

(2) Anomaly based detection**:** known as profile based detection, here a trigger is issued whenever an activity deviates from normal.

(3) Behavior-based detection**:** here suspicious behaviors are defined depending on analyzing previous history and are classified into classes

C. Signature Actions

These are the actions taken whenever an **IPS** observes any targeted action:

- Generating an alert.
- Dropping or preventing the activity.
- Logging the activity.
- Resetting a **TCP** connection.
- Blocking future activity.
- Allowing the activity.

1.3-Problem Statement

Host based **IPS** mostly depends on a static signature mechanism to identify intruders, which in turn needs to be updated from time to time to insure the most accurate detection.

To overcome the static signature detecting mechanism, we introduce in this paper a four layers host based **IPS**, which uses data mining technique, namely decision tree, as a detecting mechanism instead.

Data mining technique that will be used is the association analysis, namely the decision tree **J48** algorithm. Association analysis will be used to find out anomaly cases, where the behavior of the user is very different from the normal, which could indicate a threat.

1.4-Thesis Contribution

Since **IPS** is a new trend in security field nowadays; both network and host **IPS** are being used widely for detecting security violations. This thesis aims to produce a new technique for a host based **IPS**, that uses data mining for anomaly detection, using the J48 decision tree algorithm. This in turn uses system alerts as an input, and decides later on if there is a behavior anomaly.

Following such a procedure in **IPS**, simplifies the structure of **IPS** on one side, and contributes to the prevalence of **IPS** to be installed on each single computer in the network on the other side, which in turn duplicates the security level for the end users. Finally, depending on system alerts to reflect the up normal behavior of a computer opens the door to **IPS** to replace antivirus.

1.5-Thesis organization

The thesis consists of five chapters, each one handles a certain topic as follows: chapter one gives an introduction about the study, it treats the basic concepts in the study which are: security, data mining and **IPS**. Besides, this chapter gives a summary about the study tools like WEKA and Event viewer.

Chapter two is about the historical studies related to the thesis idea. Here the objectives of the study are being specified, and a comparison between the existing studies and the thesis is being performed, to find out similarities and differences between them.

Chapter three discusses the proposed system design and the tools used to construct the system. A briefing about each tool and the role that it plays in the system are included.

Chapter four shows the methodology used through building the proposed system, starting from choosing the system attributes to constructing the dataset file, passing through processing the file with WEKA data mining tool, and analyzing the output to build the proposed IPS actions. Also this chapter will explain the testing scenarios and list the output results.

Chapter five lists the difficulties and limitations that have affected this study and discusses the conclusion and future works.

Finally the Appendix will follow to show the code related to the study, such as decision tree **J48** algorithm, and **VB** code of the proposed **IPS**, and the input dataset file in **ARRF** format.

Chapter Two Literature Review

2.1- Introduction

Security is no longer an accessory in the modern **IT** systems, bearing in mind that, as long as communication systems are evolving and spreading, coupled with a revolution in the **IT** field, strategies of attackers will grow and get stronger everyday.

Among all security solutions that exist today, **IPS** will always be recognized as a special one. It does not need to update its database against new viruses or attacks like antivirus or firewalls, which makes it more stable and effective.

Although **IPS** is actually an **IDS** but with a difference of taking actions against inspected attacks, but this addition makes a big difference. So **IPS** is more stable, reliable and effective of all previous security solutions.

When thinking of enhancing **IPS** detecting mechanism, many things should be kept in mind. Whenever an IPS detects a problem, it will take an act, but if this was a false alert, the alert will not be taken seriously. On the other side, there will be missed alerts somehow, which means that, tuning an **IPS** is a very important issue. So if some kind of a simple mechanism could be find to make alerts connected to real practical actions, then **IPS** missed alerts will be, or say will decrease with time, and detecting rates will be so close to the reality.

Add to that, if some kind of learning technique is being added to **IPS** core, which relies on a dynamic database to identify attackers, then, the reliability and accuracy of attacking detection would be better.

To meet the first idea on enhancing **IPS** detection, anomaly detection was connected to the anomaly behavior in system resources such as memory or network traffic. This idea came from a simple fact, that any process will use system resources, so when an intruder starts to infect the system he must use a resource or more illegally, like sending packets through the network to infect new PCs, or even consume the memory, so this anomaly should indicate an intruder.

To meet the second idea, Decision tree data mining algorithm is being used in this study, as a learning technique to identifying intruders. While the previously mentioned dynamic database is being built depending on system performance, that provides a realistic entries to the decision tree algorithm.

2.2 IPS Overview

Intrusion Prevention Systems (**IPS**) are security protection devices (hardware), or applications (software), they aim to prevent attacks against network devices. These systems began life as an adjunct feature of other products, such as firewalls and antivirus products, and evolved into an independent and full featured security solution. two types of **IPS** exist: Network and Host.

When deciding where to install your Host **IPS**, the most optimum scenario to deploy Host IPS, is on every host on your network. In many situations, however, this is not achievable. This is affected by technical issues like the supported operating system, and financial ones, like budget.

Same as Host **IPS**, the most optimum protection for Network **IPS** is to deploy Network **IPS** across all the gateways of the network to inspect all of your network traffic. But it is still limited by the traffic volume on the network.

Accordingly, it is about time to overcome **IPS** installation limitations, to be able to perform the optimum protection scenarios by installing them over the network. Now the target is to simplify the detecting mechanism of Host IPS, to be able to install it over every single host on the network regardless of the operating system, and with a lower budget.

A review of related research idea is being done, in order to get use of their conclusions, and even add new things.

2.3- Data Mining and IPS

Data mining involves the use of data analysis to discover previously unknown, valid patterns and relationships in large data sets. These tools can include statistical models, mathematical algorithms, and machine learning methods. Data mining consists of more than collecting and managing data; it also includes analysis and prediction. Recently data mining started to enter information security field such as intrusion detection [14].

Intrusion detection is defined as the process of intelligently monitoring the events occurring in a computer system or network, analyzing them for signs of violations of the security policy. The primary aim of Intrusion Detection Systems (**IDS**) is to protect the availability, confidentiality and integrity of critical networked information systems [17]. Many applications of data mining techniques demonstrate that information security became an interesting field for researchers who have the tendency to discover new methods to identify the most frequent issues regarding data security in networks.

In these researches, techniques such decision trees, association rules, clustering and others are well known, and used for vulnerabilities detection.

WEKA (Waikato Environment for Knowledge Analysis), is a data mining tool that is suitable to be used to develop IPS, WEKA is a collection of machine learning algorithms for data mining tasks implemented in Java language. Weka contains tools for classification, regression, clustering, association rules, data visualization.

WEKA works with **ARFF** files (Attribute Relation File Format) and also with files in .csv format (Comma Separated Values). An **ARFF** file is a text file which describes a list of instances which share a lot of attributes. A complete description of ARFF file exists in Appendix-C

2.4- Historical Review

The following papers, used data mining techniques in **IPS** detection process:

[23] In this paper, new detection methods are developed, these methods depend on data mining techniques to discover system patterns that describe user behaviors, and use such features to compute classifications to recognize anomalies actions. Data mining algorithm used are: the association rules algorithm and the frequent episodes algorithm. Association rules are used to derive multi-feature (attribute) correlations from a database table; this means trying to determine what processes and resources are called with each other. The frequent episodes algorithm is used to collect the same data of association rules but for the process itself, it tries to identify the data and command sequence, in order to build a profile for anomaly detection. After implementing this framework on send mail system call data and network tcpdump data, it proved its effectiveness in classification models in detecting anomalies, but as long as the right feature set is chosen.

The mentioned procedure is so close to the procedure used in this study, in the matter of depending on system patterns to identify anomalies, except that, the procedure used in this study does not associate processes with their resources to decide such anomalies, it associates resources with anomalies in a predefined baseline behavior of each, in the overall environment.

[15] The goal of this paper is to characterize the normal system activities with a profile by applying mining algorithms to audit data so that abnormal intrusive activities can be detected by comparing the current activities with the profile, and as patterns of normal behavior changes with time, this paper provides an adaptive **IDS**, that works according to general framework for adaptive anomaly detection module, that uses fuzzy association rule mining. It has been observed that program executions and user activities exhibit frequent correlations among system features. Audit data can be formatted into a database table where each row is an audit record and each column is a field (system feature) of the audit records. Fuzzy association rule is used to treat audit values that have the attribute of belonging partially to more than one behavior.

The previous paper has the same base idea of this study of characterizing normal system activities in a profile, and processing it by data mining algorithms to decide abnormality, except that, **J48** decision tree algorithm is being used instead of Fuzzy algorithm in processing the built profile, and the appropriate prevention action is taken accordingly.

[21] This paper stands for using process mining techniques in analyzing audit trails to find out security threats, an audit trail is a record that stores all events in the system and the network, which keeps track between a use and system actions, which could be used in turn to relate a security violation with a user. The paper produces α-algorithm to inspect audit trails for security violations. The α-algorithm, models all acceptable behaviors and states to find out divergence from them. This algorithm could be applied at any level of security from low level to high level.

Both the previous paper and this study, have a profile that stores events related to the system and the network that will be treated later by data mining algorithm. The α-algorithm is used in the previous paper as a data mining technique, while this study uses **j48** instead.

Finally, the idea of using data mining in the detection mechanism of intruders in **IPS** was used in the previous researches, but this study uses a different data mining algorithm of them, that is **J48**, this algorithm was chosen because of its efficiency in treating missing values.

[3] This study proposes a novel **IPS** which would offer a higher level of abstraction, than the often used technique for detection which performs some processing of the contents of the network packets. Prediction are made based on patterns extracted from higher level application or operating system logs to build this specification based Intrusion Prevention System. Rather than depending on signatures, patterns of operating system calls coupled with user acts are being used to predict future patterns of anomaly. Data mining systems and algorithms are used to predict such patterns, all this will finally develop a framework for **IPS** which is an independent operating system. This paper is so close to the proposed system in this study. Both depend for the prediction of anomalies on new patterns extracted from old learned ones by using data mining.

[16] This paper raises the need for an architecture and framework specification for **IDSs**. It presents an implemented prototyped named **DAID** (Database-centric Architecture for Intrusion Detection), using the capabilities of Oracle Database. Oracle was chosen for its' known capabilities of supporting mission critical applications, distributed processing, and integration of analytics, these all made it an appropriate platform for an **IDS** implementation. The final proposed framework in this paper can be used to build, manage, deploy

, score, and analyze data mining-based intrusion detection models. The proposed Database-centric **IDS** offered many advantages over alternative systems. These include tight integration of individual components, security, scalability, and high availability.

[8] In this paper data mining techniques are used for intrusion detection. Data provided by National Vulnerability Database site represents the basic input for the application and the goal is to demonstrate that the proposed system could identify appropriate and accurate classifiers to detect anomalies mediated by data mining methods, namely association rules and decision trees. this paper works on a methodology that is very close to this study. except that the input dataset is being constructed from the scratch in this study.

2.5 The Study versus Historical Reviews

After going through the previous historical review, the paragraph below summarizes the similarities and differences between previous papers and this study.

This study uses system resources as an input attributes for **J48** algorithm, it depends on system alerts to report for any odd behavior in the system. This study does not relate resources to the user or to a specific process, like the previous papers; indeed it relates resources to a pre defined baseline measure for each resource. This procedure is actually too close to the above, in the idea of finding a real measurements or indicators for anomalies from the system itself, but this study still has a distinguished technique, that plays a key role in decreasing false positive alarms and false negative alarms, that have been mentioned previously, by providing a simple feedback to commit the action taken by **IPS** in order to tune it correctly with time.

The following chapter is about tools and technologies used in this study.

Chapter tree
proposed system tools

3.1- Introduction

The researcher aims to achieve an **IPS** that meets the conditions previously mentioned, of simple detecting mechanism, lower installation cost, and compatibility with any operating system. This Chapter shows the proposed system design, and describes in details the tools used in building up the system.

3.2- The Proposed System Design

The system was designed in the form of four logical layers; each layer plays a certain role, and these layers integrate with each others to produce the whole complete system, eventually. A specified technology was used in each layer when developing the system, in order to meet its rule.

The first layer is the Client layer; it contains the user interface controls, which in turn takes the role of communicating with all the code behind. VB.Net technology was used as a developing environment tool for this layer.

Then comes the Business Logic Services layer, this layer will handle data mining algorithm and hands the output to Data Services layer. WEKA the famous data mining tool was used to perform **J48** association algorithm on the input data at this layer.

Now, the Data Services Layer has two jobs indeed, it takes the appropriate action that will be taken by the system whenever an intruder is being detected, and alarm the user about the suspected action. The first job is the responsibility of the Prevention Module of that layer. While the second job, is the responsibility of the Alarm Module. This module depends on Business Logic Services layers' decisions to start its job. ASP.Net technology was used as a developing environment tool at this layer.

The Data layer contains the input data for the Business Logic Services layer, which is populated through system alerts. It is being updated through a feedback sent from Data Service Layer. Comma separator file format is being used at that layer.

Figure 3.1 demonstrates system architecture, and application interaction through the network.

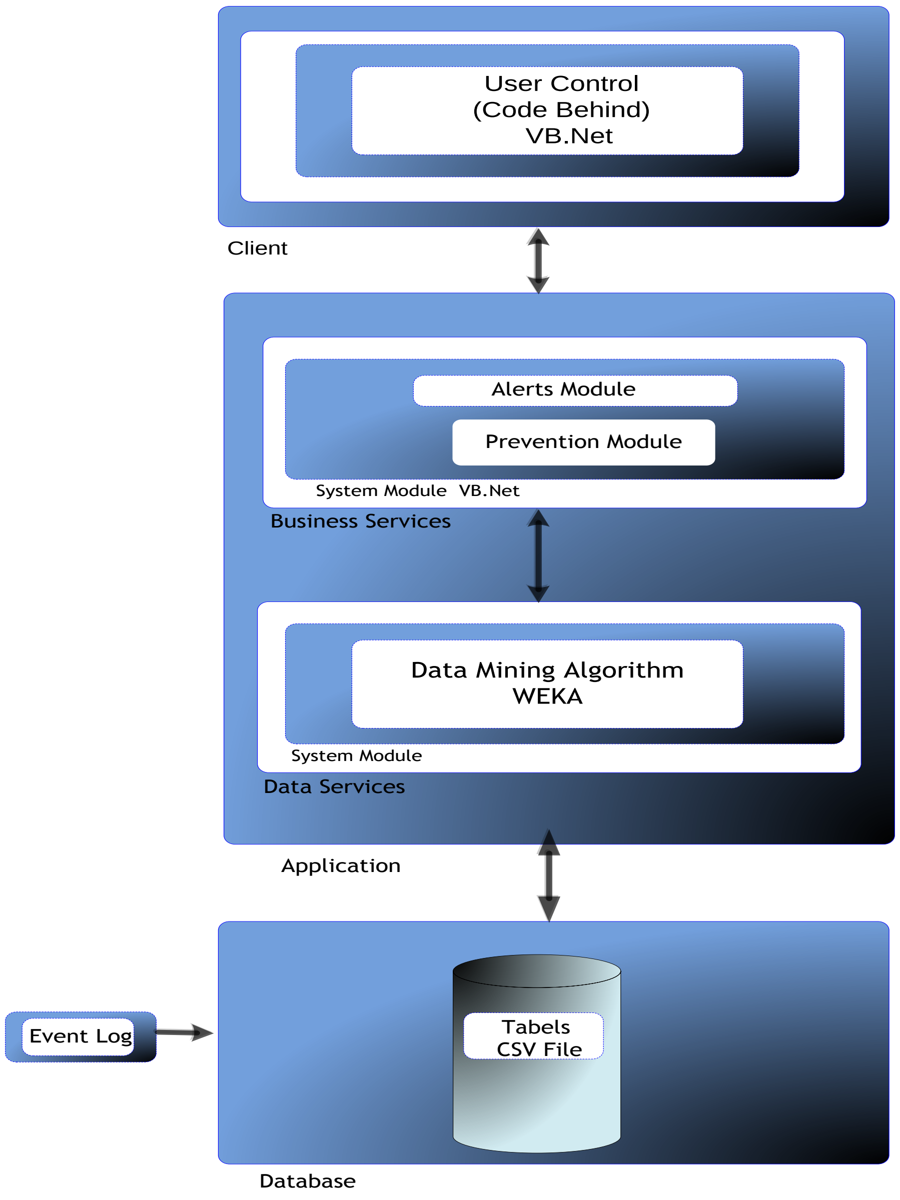


Figure 3.1- System Architecture

Figure 3.2 shows system interaction in the network, the intruder can gain access to the network using wireless technology or through the internet, with the existence of the proposed **IPS** installed on top of each local **PC** the level of security will be enhanced.

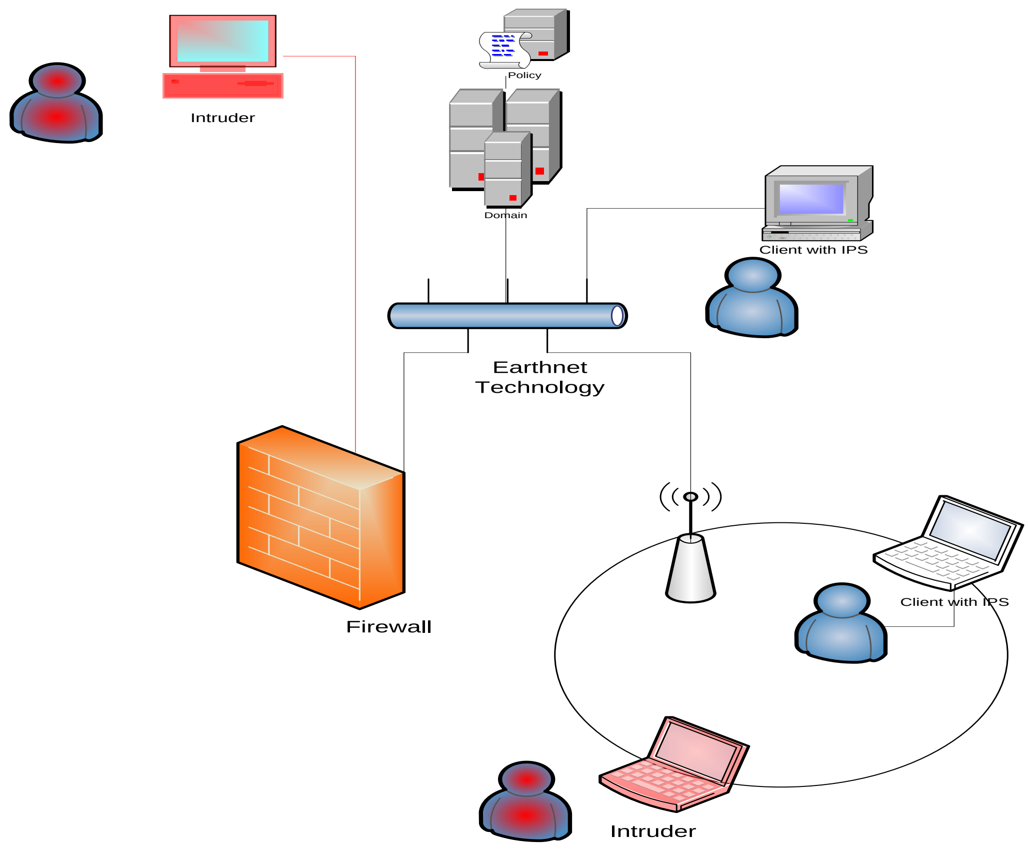


Figure 3.2- Application Interaction

3.3- Used Tools

Now, the tools used to develop the system are being discussed in details.

3.3.1- .NET Technology

.NET is Microsoft’s platform for building **XML** Web services. With **.NET**, Microsoft formalizes a vision of an Internet made up of an infinite number of interoperable Web applications or services, which will operate in concert to form a global exchange network. The **.NET** Framework is really a strategy to tie disparate platforms and devices together, moving data around in a far more efficient manner than it is currently.

.NET is Microsoft's platform for Web Services. Web Services allow applications to communicate and share data over the internet, regardless of the operating system or programming language. The Microsoft **.NET** platform includes a comprehensive family of products, built on internet standards such as **XML** and **HTTP**, that provide facilities for developing, managing, using, and experiencing **XML** Web services. There are five areas where Microsoft builds the **.NET** platform: **.NET** Experiences, Clients, Services, Servers, and Tools.

Visual Studio **.NET** and the Microsoft **.NET** Framework supply a complete solution for developers to build, deploy, and manage Web Services. The **.NET** Tools maximize the performance, reliability, and security of Web Services. Visual Studio **.NET** is the next generation of Microsoft's multi-language development environment.

Visual Studio **.NET** will help developers quickly build Web Services and applications (including **ASP.NET** applications) using their language of choice. Visual Studio **.NET** advances the high-productivity programming languages

Visual Basic, which includes new Object-oriented programming features; Visual **C++**, which advances Windows development and enables you to build **.NET** applications; and **C#** (pronounced C sharp).

The **.NET** Framework is a high-productivity, standards-based, multi-language application execution environment that handles the essential “house keeping” chores and eases deployment and management. It provides an application execution environment that manages memory, addresses, versioning issues, and improves the reliability, scalability, and security of applications. The framework consists of several parts, including the Common Language Runtime and **ASP.NET**.

ASP is relatively new Web development technology. Although it is very powerful and simple to use, it has some flaws. With **ASP.NET**, Microsoft has introduced a new Web development Platform that addresses many, if not all, of **ASP**’s shortcomings. **ASP.NET** offers many Programmatic improvements including a new data access technology called **ADO.NET**.

ADO.NET is designed to work on the Web, which is inherently disconnected. **ASP.NET** and **ADO.NET** are part of larger framework, generically referred to as the **.NET**

Framework. The **.NET** Framework is a set of products and services designed to facilitate the development of interoperable Web applications based on open standards such as **XML**, and **HTTP**.

3.3.2- Data Mining and WEKA

Data mining is about analyzing data and finding hidden patterns using automatic or semiautomatic means [24]. Data mining extracts new unknown useful information from data by using certain techniques that can be applied to many applications, answering various types of businesses questions. Data Mining can perform the following tasks on data:

- *Clustering*, which refers to classifying cases into categories depending on a certain attribute?
- *Classifying*, it performs natural groupings based on a set of attributes.
- *Association*, is to identify common sets of items and rules for the problem.
- *Regression*, this is similar to classifying process but for continuous values.
- *Forecasting*, this is used to predict certain attribute statistics.

Many data mining tools are available today, such as WEKA, YALE, SQL Server 2005, and matlab. WEKA is known as a powerful workbench in data mining. It is mainly used in education field.

The WEKA stands for Waikato Environment for Knowledge Analysis; it was developed at the University of Waikato in New Zealand. WEKA workbench is a collection of state-of-the-art machine learning algorithms and data preprocessing tools [Ian05]. It provides all the processes of data mining techniques, starting from preparing the data, visualizing both inputs and outputs, and finally evaluating the learning algorithms. WEKA has the logo of a flightless bird that is found only on the islands of New Zealand Called Mecca, this logo came from the rhyming of WEKA and Mecca.

A.WEKA Components

WEKA implements learning algorithms that could be applied to a dataset, it includes methods for handling most data mining problems such as: regression, classification, clustering, association rule mining, and attribute selection.

WEKA learning methods are called classifiers, which can be used in one of the following ways: applied to a dataset to analyze the output and learn about the data, or generate predictions on new instances or compare classifiers performance in order to choose the best prediction.

These Classifiers has parameters that could be tuned through object editor. WEKA also has what are called filters, which are tools for processing the data. This workbench also has algorithms for association rules and data clustering.

B.WEKA Interface

WEKA has four methods to treat data; this depends on the target of the experiment itself. Those methods can be accessed by the Explorer button, or Simple **CLI** (Command Line Interface) button, or Experimenter button or knowledge flow button. Figure 3.3 shows the main Graphical User Interface (**GUI**) that appears once you choose WEKA software.

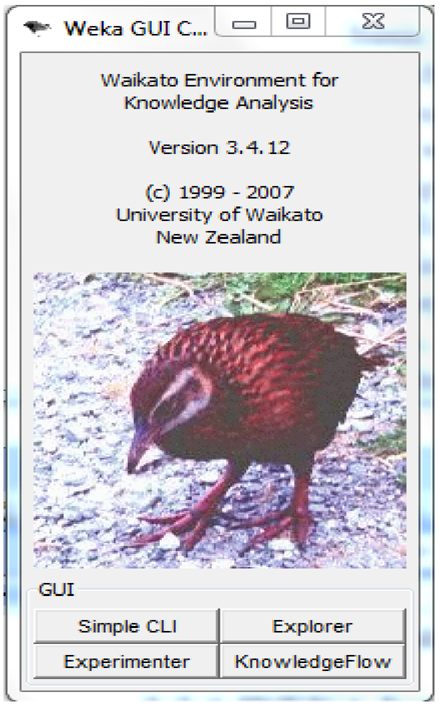
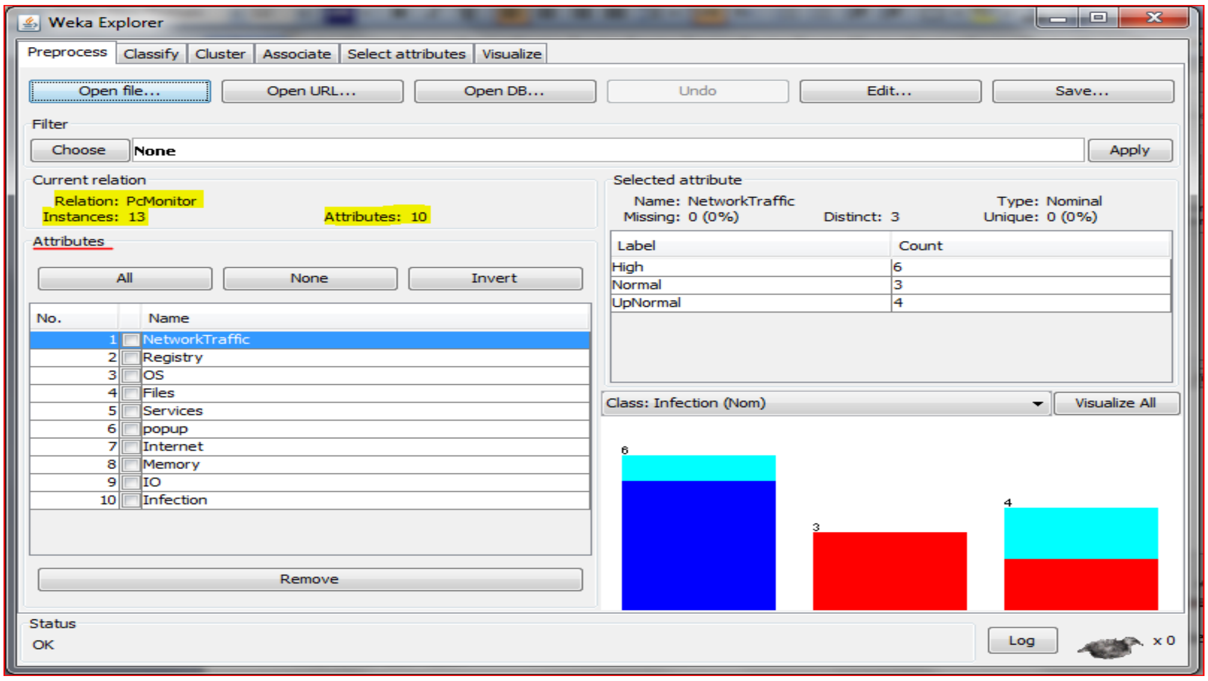


Figure 3.3- WEKA GUI

Explorer **GUI**, is the most convenient way to process your data, it gives access to all of its facilities using menu selection and form filling. But it has a disadvantage of holding everything in the main memory, which limits its activity to medium sized experiments. Figure 3.4 show the explore interface.

The first tab is the Preprocess tab. Any dataset can be loaded using this tab. The marked tags in this window show the following: relation name which is PCMonitor, number of instances (records in the dataset, which are 13) and the number of attributes (which are 10), all these attributes are shown in the left pan (under the red underlined word attribute) where the columns in the right show the relation between the attributes.



None

Current relation
Relation: PcMonitor
Instances: 13
Attributes: 10

Selected attribute
Name: NetworkTraffic
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)

Label	Count
High	6
Normal	3
UpNormal	4

Attributes

No.	Name
1	NetworkTraffic
2	Registry
3	OS
4	Files
5	Services
6	popup
7	Internet
8	Memory
9	IO
10	Infection

Class: Infection (Nom)

Status
OK

Figure 3.4- Explorer GUI-Preprocess

To process the dataset, the classify tab is used to treat the dataset using different data mining algorithms, figures 3.5 and 3.6 show how to select a classifier and the output results.

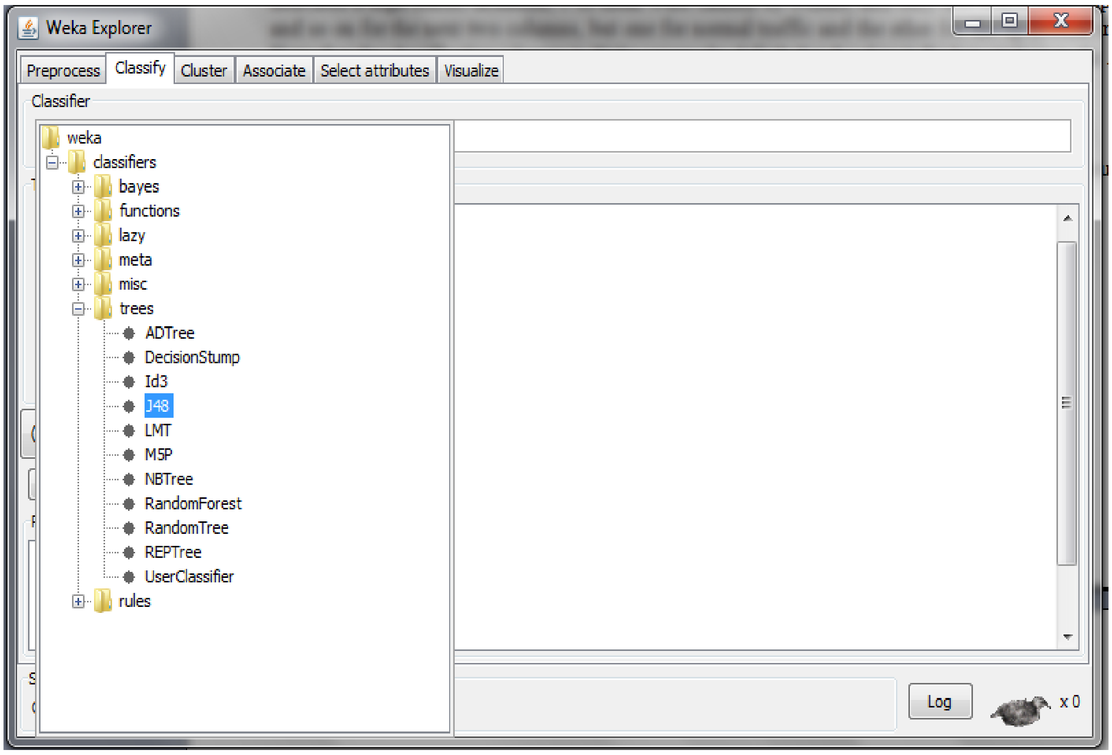
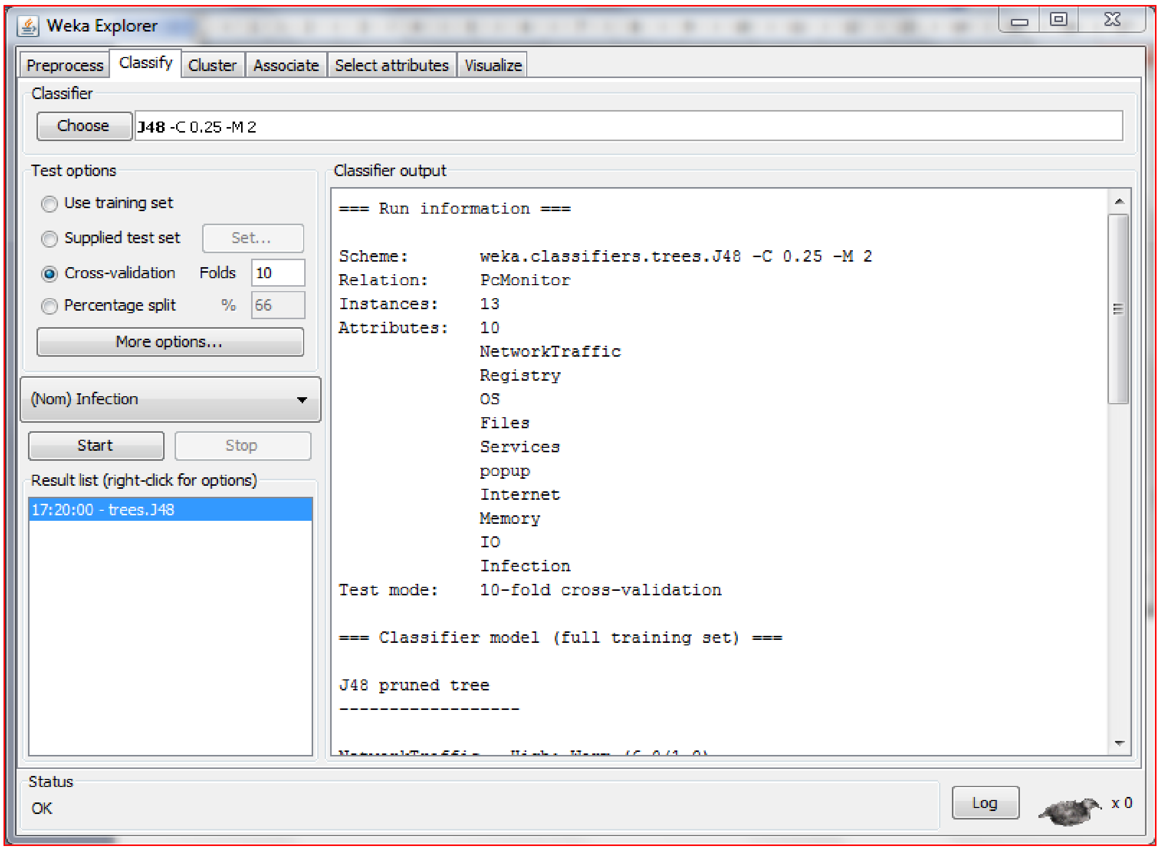


Figure 3.5- Explorer GUI-Classify



J48 -C 0.25 -M 2

Use training set
Supplied test set
[✓ Cross-validation] Folds 10
Percentage split % 66

(Nom) Infection

17:20:00 - trees.J48

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     PcMonitor
Instances:    13
Attributes:   10
              NetworkTraffic
              Registry
              OS
              Files
              Services
              popup
              Internet
              Memory
              IO
              Infection
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------
```

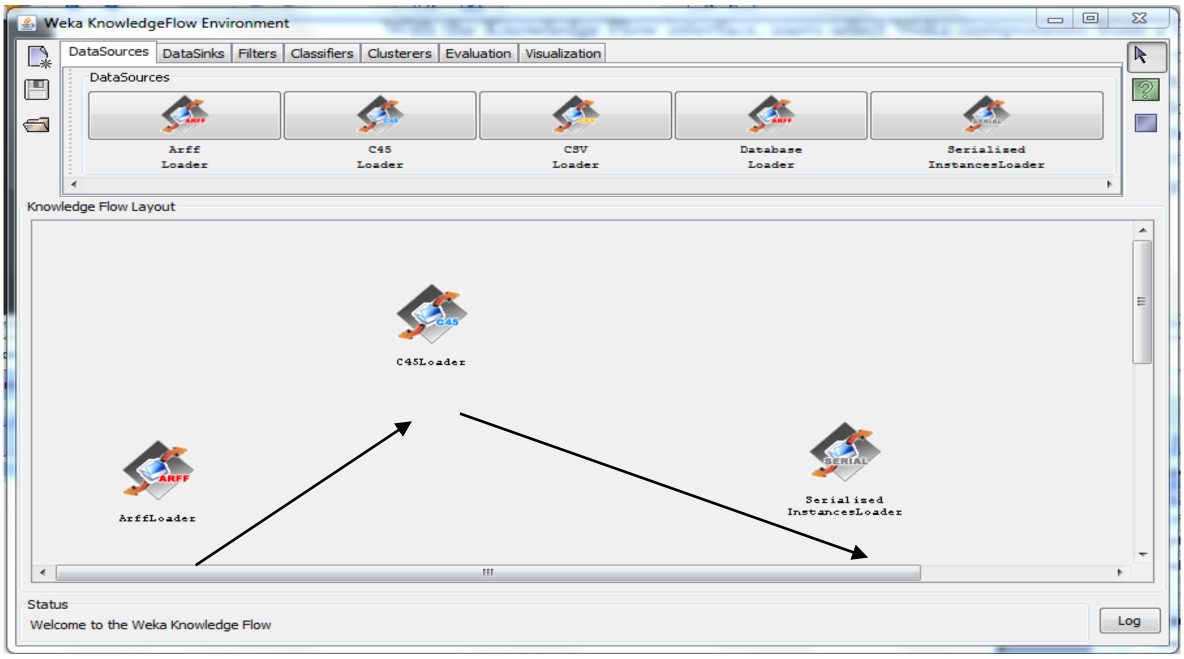
Status
OK

Figure 3.6- Explorer GUI-Classify Results

Note: a complete demonstration of **J48** algorithm and the shown results will be listed in the Appendix.

The rest of the tabs are: Cluster that Learns clusters for the dataset, associate that learns association rules for the data and evaluate them, Select attributes: Select the most relevant aspects in the dataset and visualize different two-dimensional plots of the data and interact with them.

Knowledge Flow **GUI,** allows you to design configurations for streamed data processing by dragging boxes representing learning algorithms and data sources around the screen and join them together into the wanted configuration. Figure 3.7 shows the **GUI** for the knowledge flow.

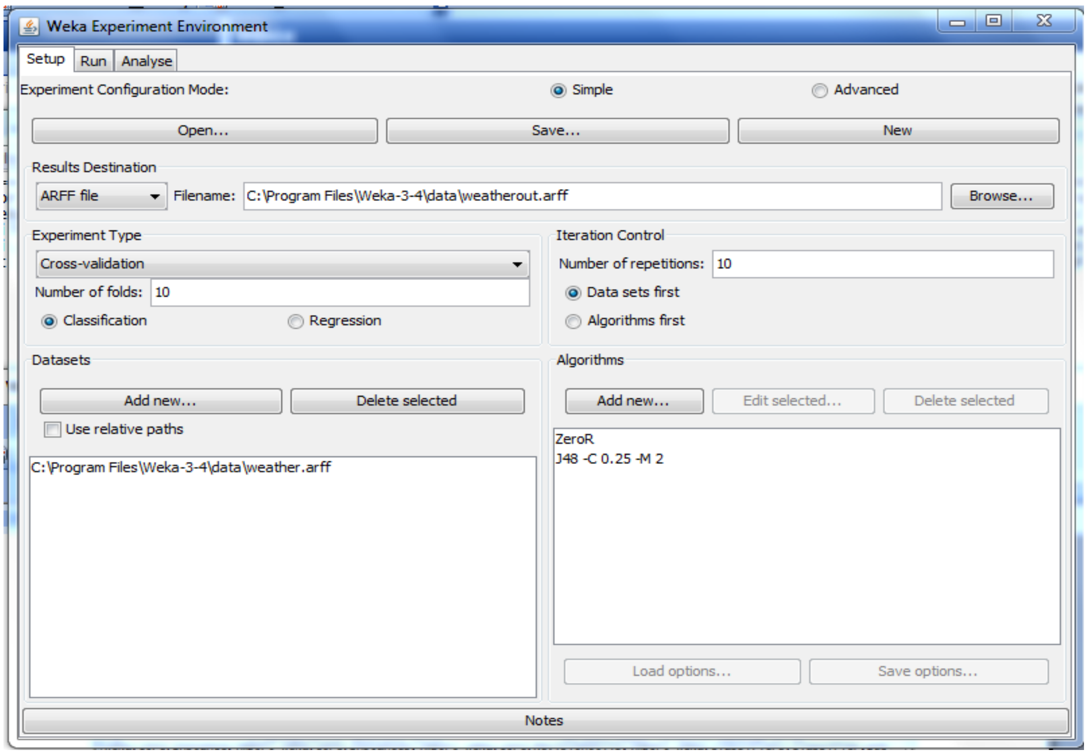


Welcome to the Weka Knowledge Flow

Figure 3.7- Knowledge Flow GUI

Experimenter **GUI**, is designed to help answering basic practical question when applying classification and regression techniques. It provides the environment to compare a variety of learning techniques. Figures 3.8 shows the experimenter **GUI**.

This **GUI** has the following three tabs: Setup tab, it is used to load the dataset and select more than one learning algorithm to compare their results and export the results to a specified file. Run tab, which is used to run the process. Analyse tab, that shows the results of the comparison in the **GUI** itself. Figure 3.9 shows the output file for the experimenter.



Experiment Configuration Mode: [✓ Simple] Advanced

Results Destination
ARFF file Filename: C:\Program Files\Weka-3-4\data\weatherout.arff

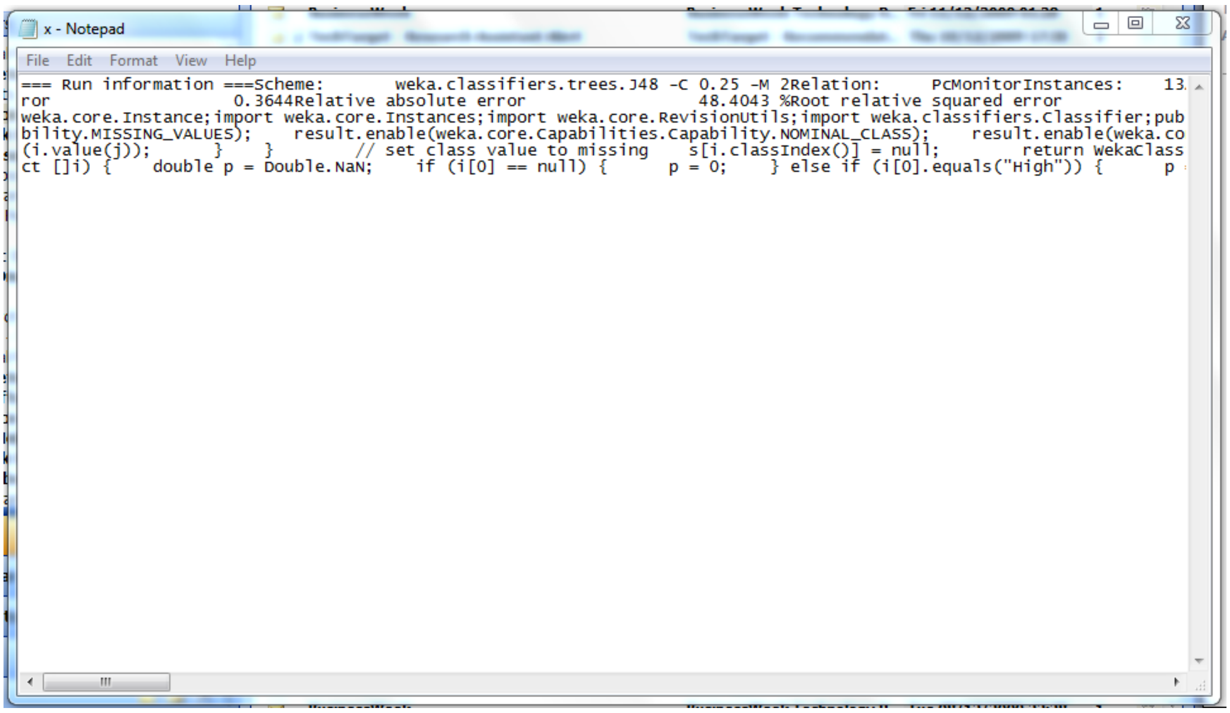
Experiment Type
Cross-validation
Number of folds: 10
[✓ Classification] Regression

Iteration Control
Number of repetitions: 10
[✓ Data sets first]
Algorithms first

Datasets
Use relative paths
C:\Program Files\Weka-3-4\data\weather.arff

Algorithms
ZeroR
J48 -C 0.25 -M 2

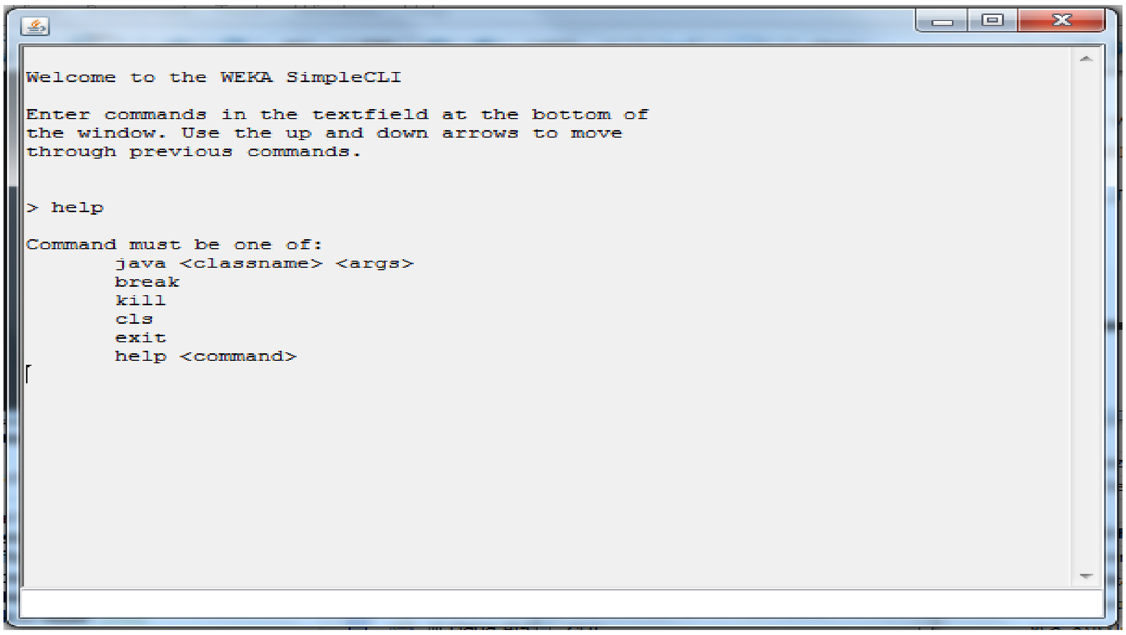
Figure 3.8- Experiment Environment GUI



```
=== Run information ===Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2Relation:     PcMonitorInstances:    13.
ror                  0.3644Relative absolute error                   48.4043 %Root relative squared error
weka.core.Instance;import weka.core.Instances;import weka.core.RevisionUtils;import weka.classifiers.Classifier;pub
bility.MISSING_VALUES);    result.enable(weka.core.Capabilities.Capability.NOMINAL_CLASS);    result.enable(weka.co
(i.value(j));       }    }        // set class value to missing    s[i.classIndex()] = null;        return WekaClass
ct []i) {    double p = Double.NaN;    if (i[0] == null) {      p = 0;    } else if (i[0].equals("High")) {      p
```

Figure 3.9- Output File-Experiment Environment GUI

Simple Command Line Interface (**CLI**), unlike all of above, presents access to the WEKA code, everything here should be performed by typing up the appropriate command. Figure 3.9 shows the GUI for the command line.



```
Welcome to the WEKA SimpleCLI

Enter commands in the textfield at the bottom of
the window. Use the up and down arrows to move
through previous commands.


> help

Command must be one of:
        java <classname> <args>
        break
        kill
        cls
        exit
        help <command>
```

Figure 3.9- Output File-Experiment Environment GUI

3.4- Performance Monitor and System Alerts

Windows has a powerful tool to monitor computer performance and generate alerts that is the Performance Monitor tool found in the control panel. Using this tool any resource could be monitored i.e RAM, memory I/O, network traffic and many else, alarms warn the user about any problem related to the monitored resource. In Windows 7.0 the output of performance monitor can be exported to **XML** view so it could be handled easily by other applications. Figure 3.10 shows the performance monitor tool. To open it, go to start/all programs/administrator tools /performance.

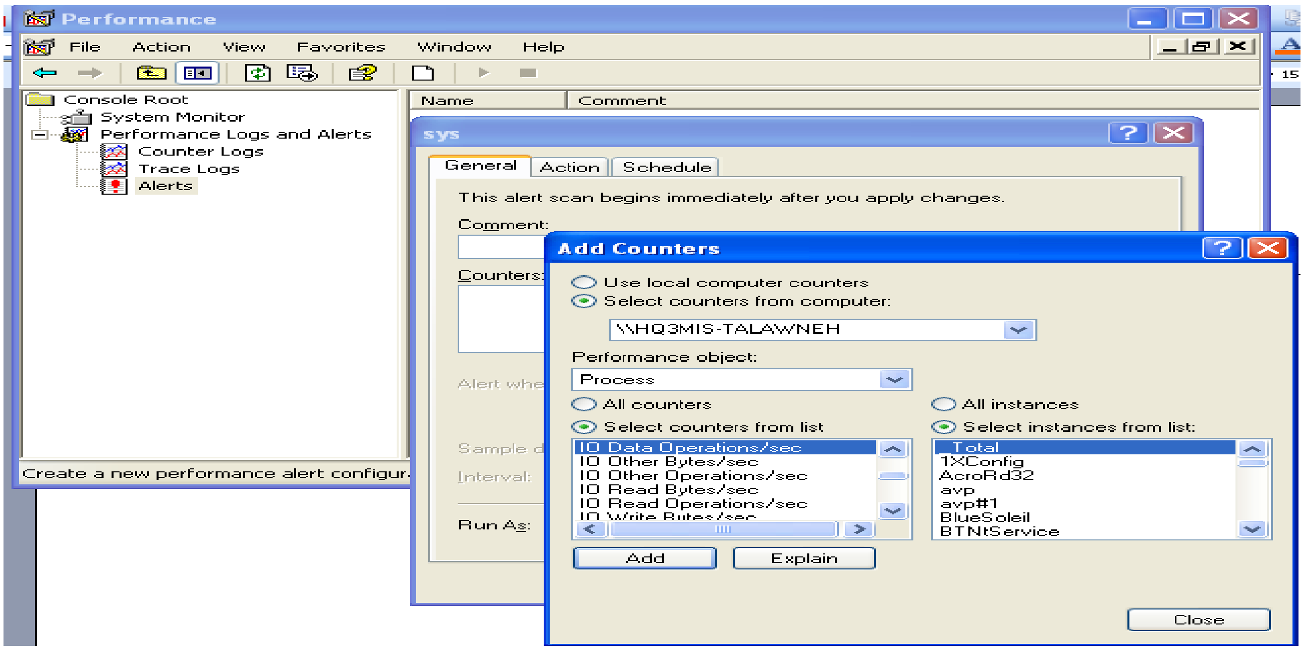
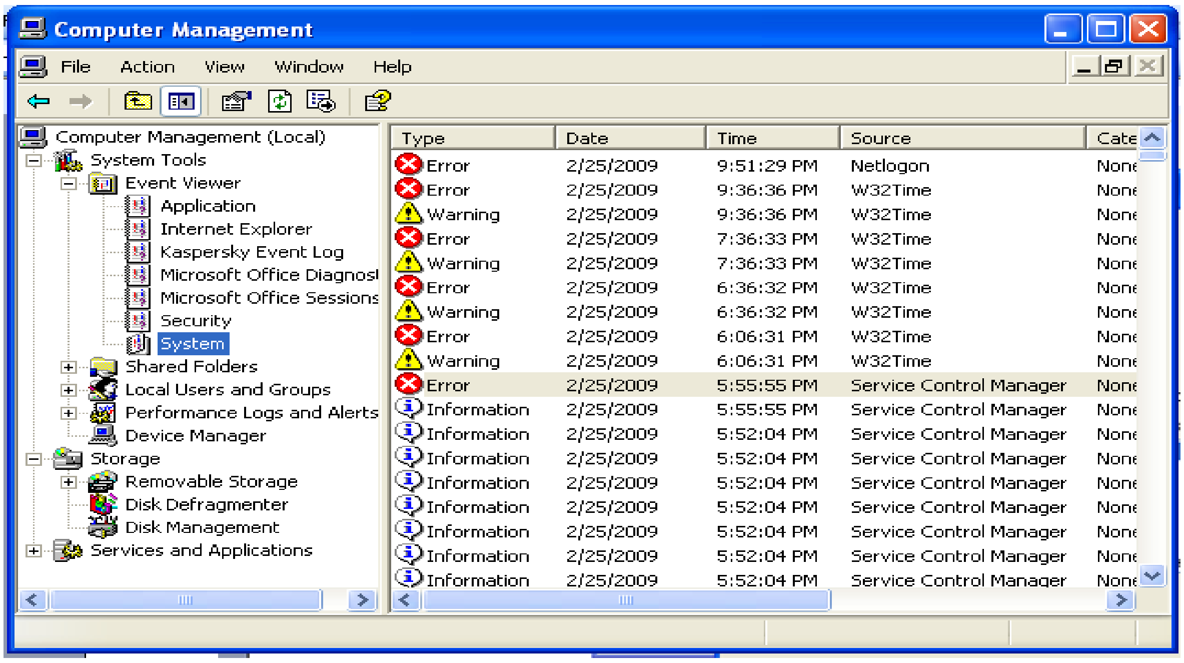


Figure 3.10- Performance Monitor

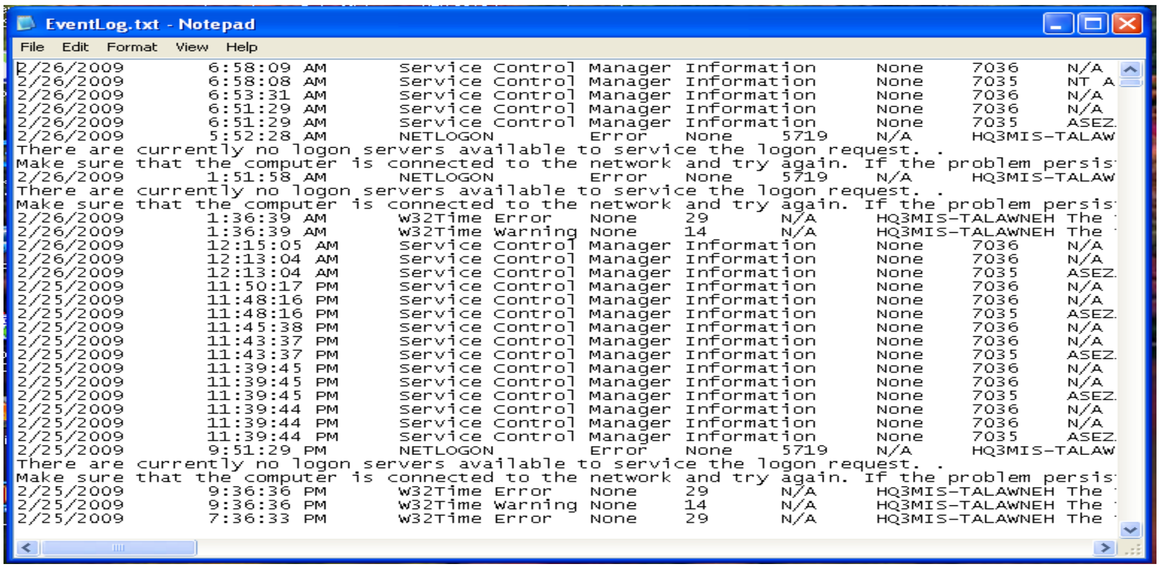
Any alarm of the built counters will be logged in the system event viewer as shown in figure 3.11.



Type	Date	Time	Source	Cate
Error	2/25/2009	9:51:29 PM	Netlogon	None
Error	2/25/2009	9:36:36 PM	W32Time	None
Warning	2/25/2009	9:36:36 PM	W32Time	None
Error	2/25/2009	7:36:33 PM	W32Time	None
Warning	2/25/2009	7:36:33 PM	W32Time	None
Error	2/25/2009	6:36:32 PM	W32Time	None
Warning	2/25/2009	6:36:32 PM	W32Time	None
Error	2/25/2009	6:06:31 PM	W32Time	None
Warning	2/25/2009	6:06:31 PM	W32Time	None
Error	2/25/2009	5:55:55 PM	Service Control Manager	None
Information	2/25/2009	5:55:55 PM	Service Control Manager	None
Information	2/25/2009	5:52:04 PM	Service Control Manager	None
Information	2/25/2009	5:52:04 PM	Service Control Manager	None
Information	2/25/2009	5:52:04 PM	Service Control Manager	None
Information	2/25/2009	5:52:04 PM	Service Control Manager	None
Information	2/25/2009	5:52:04 PM	Service Control Manager	None
Information	2/25/2009	5:52:04 PM	Service Control Manager	None
Information	2/25/2009	5:52:04 PM	Service Control Manager	None

Figure 3.11 System Alarms

Alerts in event viewer, has the ability to be exported in a text format, which makes it easy to access the file from another program, to read certain values. Figure 3.12 shows system alerts in text format.



```
2/26/2009        6:58:09 AM      Service Control Manager Information     None     7036     N/A
2/26/2009        6:58:08 AM      Service Control Manager Information     None     7035     NT A
2/26/2009        6:53:31 AM      Service Control Manager Information     None     7036     N/A
2/26/2009        6:51:29 AM      Service Control Manager Information     None     7036     N/A
2/26/2009        6:51:29 AM      Service Control Manager Information     None     7035     ASEZ.
2/26/2009        5:52:28 AM      NETLOGON        Error   None    5719    N/A      HQ3MIS-TALAW
There are currently no logon servers available to service the logon request. .
Make sure that the computer is connected to the network and try again. If the problem persis
2/26/2009        1:51:58 AM      NETLOGON        Error   None    5719    N/A      HQ3MIS-TALAW
There are currently no logon servers available to service the logon request. .
Make sure that the computer is connected to the network and try again. If the problem persis
2/26/2009        1:36:39 AM      W32Time Error   None    29      N/A     HQ3MIS-TALAWNEH The
2/26/2009        1:36:39 AM      W32Time Warning None    14      N/A     HQ3MIS-TALAWNEH The
2/26/2009        12:15:05 AM     Service Control Manager Information     None     7036     N/A
2/26/2009        12:13:04 AM     Service Control Manager Information     None     7036     N/A
2/26/2009        12:13:04 AM     Service Control Manager Information     None     7035     ASEZ.
2/25/2009        11:50:17 PM     Service Control Manager Information     None     7036     N/A
2/25/2009        11:48:16 PM     Service Control Manager Information     None     7036     N/A
2/25/2009        11:48:16 PM     Service Control Manager Information     None     7035     ASEZ.
2/25/2009        11:45:38 PM     Service Control Manager Information     None     7036     N/A
2/25/2009        11:43:37 PM     Service Control Manager Information     None     7036     N/A
2/25/2009        11:43:37 PM     Service Control Manager Information     None     7035     ASEZ.
2/25/2009        11:39:45 PM     Service Control Manager Information     None     7036     N/A
2/25/2009        11:39:45 PM     Service Control Manager Information     None     7036     N/A
2/25/2009        11:39:45 PM     Service Control Manager Information     None     7035     ASEZ.
2/25/2009        11:39:44 PM     Service Control Manager Information     None     7036     N/A
2/25/2009        11:39:44 PM     Service Control Manager Information     None     7036     N/A
2/25/2009        11:39:44 PM     Service Control Manager Information     None     7035     ASEZ.
2/25/2009        9:51:29 PM      NETLOGON        Error   None    5719    N/A      HQ3MIS-TALAW
There are currently no logon servers available to service the logon request. .
Make sure that the computer is connected to the network and try again. If the problem persis
2/25/2009        9:36:36 PM      W32Time Error   None    29      N/A     HQ3MIS-TALAWNEH The
2/25/2009        9:36:36 PM      W32Time Warning None    14      N/A     HQ3MIS-TALAWNEH The
2/25/2009        7:36:33 PM      W32Time Error   None    29      N/A     HQ3MIS-TALAWNEH The
```

Figure 3.12 System Alerts Text File

After discussing the tools used to build the **IPS** system, it is time to explain in details the construction of the system. The following chapter gives a detailed explanation about the proposed system construction.

Chapter four
proposed system methodology

4.1- Introduction

When describing the methodology used to build the system, we must refer back to Figure 3.1 in the previous chapter, which shows the four logical layers that compose the System Architecture. Each logical layer was developed from a collection of several steps depending on a certain technology or tool. Now to build these layers the following phases were performed.

4.2 - Building System Performance Alerts

To build the lowest logical layer that appears in figure 3.1, the database layer, the following steps were performed:

1. Specify system resources to be monitored.
2. Monitor the performance of selected resources for a period of time that would reflect the normal operation (threshold) of each.
3. Build system alerts based on the previous thresholds.

1. Specify system resources to be monitored. After reviewing the most famous attacks (security holes, viruses, Trojan horses, warms, spaywors …) targeting computers in the last five years, certain symptoms were noticed in common, that indicate an up normality in the performance of the infected machine. This led to the idea of monitoring the most attacked resources in computers and taking an

2. action once an anomaly behavior is noticed in any.

Table 4.1 shows the top 10 attacks in this year and their related symptoms, depending on that table, the computer resources that should be monitored closely were decided, they were: network interface card, services, file modification, registry, memory, internet and input output actions.

These resources can be monitored using a system performance tool that exists in the control panel under the administrator tools category. But to do that, there are many counters for each single resource, which means that certain counters should be selected under the condition that these counters should represent our demands. Table 4.2 shows these counters, the procedure to create such counters was treated in details in section 3.3- Performance Monitor and System.

Table 4.1- Top 10 Virus Attacks in 2009

Name	**Type**	**Vulnerability**	**Network**	**Services**	**Credentials**	**Memory**	**File modification**	**I/O**	**Internet**	**Hardware problem**	**Popup windows**
Win32/Conficker (a,b,c,d,e)	Network worm	RPC in MS OS	High traffic	Disable windows update & antivirus	Lock username						
INF/Autorun (W32.Bacalid)	worm		High traffic			Consume resources	Change in registry	High			
Win32/PSW	Trojan					Consume resources (PC slow)			slow	AVG slow	yes
Wind32/Agent	Trojan						Change in registry		no connection & redirect & download		yes
Win32/FlyStudio	worm		High traffic			PC slow	OS files				yes
INF/Conficker	worm		High traffic			PC slow		high	Slow & Change settings & redirect		yes

Win32/Pacex.Gen	trojan					PC slow			Slow & Change settings & redirect		
WMA/TrojanDownloader.GetCodec	trojan					PC slow			slow after downloading a video		
Win32/Qhost	trojan						Change in registry		Slow & Change settings & redirect		yes
Win32/Autorun	worm		High traffic			Pc slow	Chang in registry & deleting files				

(http://www.symantec.com/connect/blogs/top-10-computer-viruses-2009

Table 4.2- Monitored Resources Counters

Resource	Counter/Sub counters		Meaning
Network Interface Card	Network Interface	Bytes Received/sec	the rate at which bytes are received over each network adapter
		Bytes Sent/sec	the rate at which bytes are sent over each network adapter
	Redirector (performance object consists of counter that monitor network connections originating at the local computer)	Bytes Received/sec	the rate of bytes coming in to the Redirector from the network. It includes all application data as well as network protocol information (such as packet headers)
		Bytes Transmitted/sec	the rate at which bytes are leaving the Redirector to the network. It includes all application data as well as network protocol information (such as packet headers and the like)
Services	Security Systems Wide Statistics (these counters track authentication performance on a per second basis)	KDS As Requested	This counter tracks the number of Authentication Service (AS) requests that are being processed by the Key Distribution Center (KDC) per second. Clients use AS requests to obtain a ticket-granting ticket
File Modification	Search gathering Project (Counters for the Windows Search Service Gatherer Project object)	Changed Doc	Documents which have changed since the last crawl (system slow)
		Document delete rate	The number of document deletes per second.
		Document Modify rate	The number of modify notifications per second

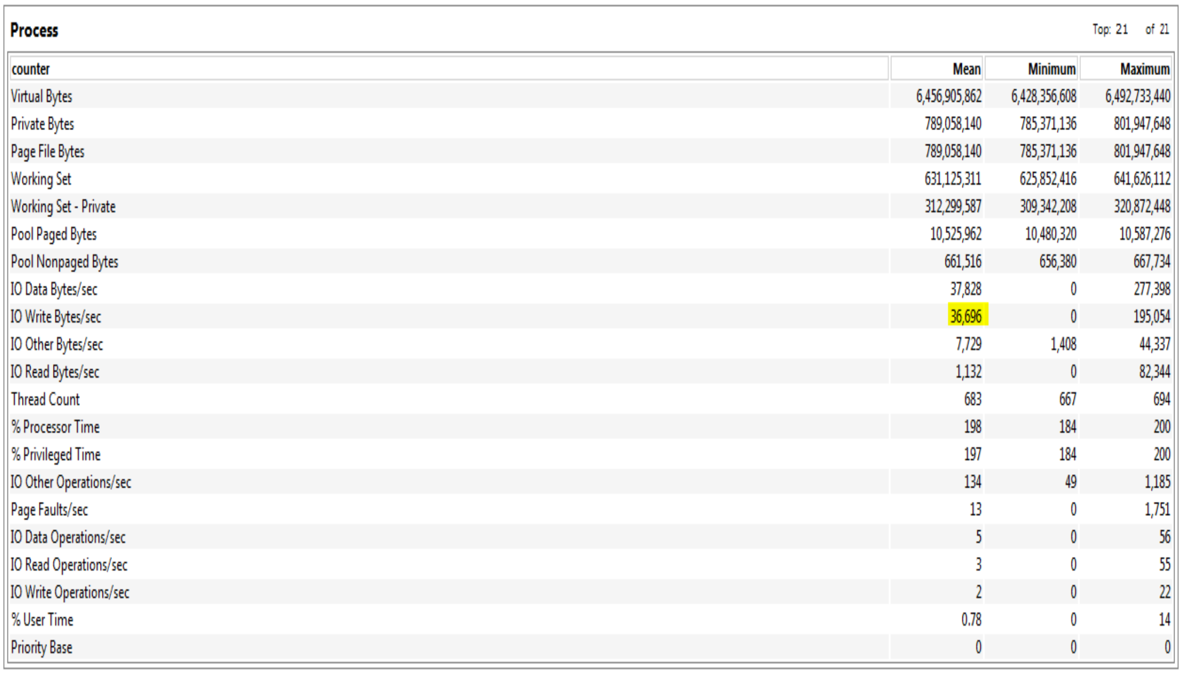
	Process	Process IO write operation /sec	The rate at which the process is issuing write I/O operations. This counter counts all I/O activity generated by the process to include file, network and device I/Os
Registry Files	System	% Registry Quota In Use	the percentage of the Total Registry Quota Allowed that is currently being used by the system
Internet	Search Gathering (Counters for the Windows Search Service Gathering service object)	Filter object	The number of filter objects in the system. Each filter object corresponds to a URL currently being filtered
	The Internet Information Services Global (includes counters that monitor Internet Information Services (the Web service and the FTP service) as a whole)	Current file cached	Current number of files whose content is in the cache for WWW and FTP services

Resources	Memory & IO	Cache	The Cache performance object consists of counters that monitor the file system cache, an area of physical memory that stores recently used data as long as possible to permit access to the data without having to read from the disk. Because applications typically use the cache, the cache is monitored as an indicator of application I/O operations. When memory is plentiful, the cache can grow, but when memory is scarce, the cache can become too small to be effective
		LogicalDisk- Avg. Disk Bytes/Transfer	is the average number of bytes transferred to or from the disk during write or read operations
		System- Processor Queue Length	Processor Queue Length is the number of threads in the processor queue. Unlike the disk counter
		System performance level	Indicates the level of the amount of system resources that the Gatherer service is allowed to use

3. Monitor the performance of selected resources. Now the previous resources are being monitored for a period that is quite enough to reflect system normal operation (Finding the baseline for each resource), luckily to have Windows 7.0 administrator tools/performance tool, have the ability to export the report of resource monitoring in an **XML** format.

4. Also it computes the mean for each resource counter. Having these two facilities, that were not available in the previous versions of windows, made it so easy to find the base line operation level for the monitored resources. Calculating the mean value of operation for each resource would be a big problem with the absence of these added features of Windows 7.0. Table 4.3 below shows a performance report for a Process entry, the mean is shown for every process counter.

Table 4.3- Performance Report for Process Entry

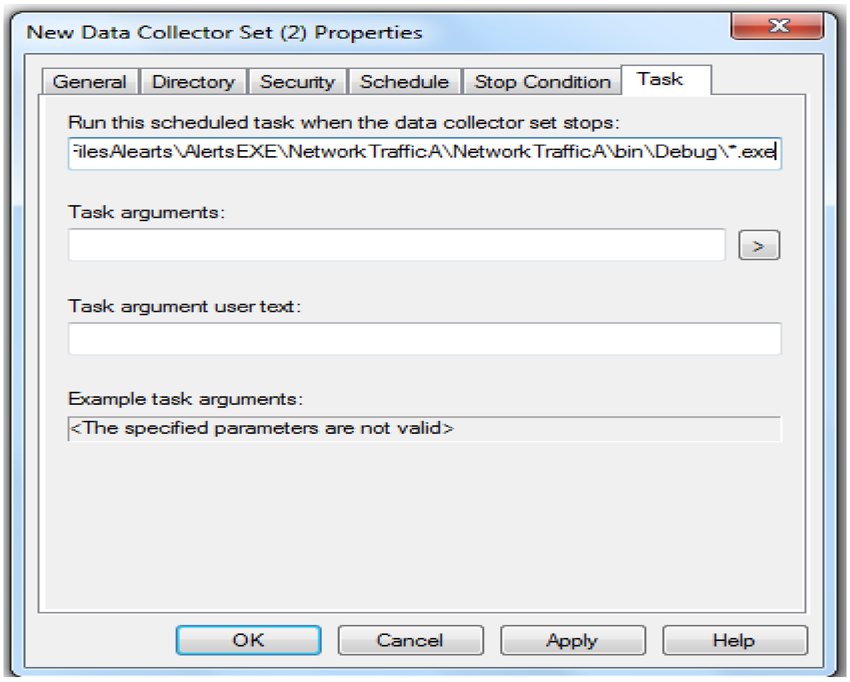


Process Top: 21 of 21

counter	Mean	Minimum	Maximum
Virtual Bytes	6,456,905,862	6,428,356,608	6,492,733,440
Private Bytes	789,058,140	785,371,136	801,947,648
Page File Bytes	789,058,140	785,371,136	801,947,648
Working Set	631,125,311	625,852,416	641,626,112
Working Set - Private	312,299,587	309,342,208	320,872,448
Pool Paged Bytes	10,525,962	10,480,320	10,587,276
Pool Nonpaged Bytes	661,516	656,380	667,734
IO Data Bytes/sec	37,828	0	277,398
IO Write Bytes/sec	36,696	0	195,054
IO Other Bytes/sec	7,729	1,408	44,337
IO Read Bytes/sec	1,132	0	82,344
Thread Count	683	667	694
% Processor Time	198	184	200
% Privileged Time	197	184	200
IO Other Operations/sec	134	49	1,185
Page Faults/sec	13	0	1,751
IO Data Operations/sec	5	0	56
IO Read Operations/sec	3	0	55
IO Write Operations/sec	2	0	22
% User Time	0.78	0	14
Priority Base	0	0	0

5. Build system alerts. The mean value of each monitored resource is used as the triggering threshold when building their related system alarms. This means that, whenever a resource exceeds this value, a system alert would be triggered. There is also a useful feature in the system alerts tool; it has the ability to run a program whenever an alert occurs. Figure 4.1 shows this feature.

This feature was used to fill the input database file named AlertsDataset with its' entries, a simple executable format **VB** program was written for each monitored resource, its job is to lo that resource alert to the input database file. Figure 4.1 below shows the call of one of these **VB** programs that logs the alert of the network traffic.


New Data Collector Set (2) Properties

Run this scheduled task when the data collector set stops:
FilesAlearts\AlertsEXE\NetworkTrafficA\NetworkTrafficA\bin\Debug\*.exe

Task arguments:

Task argument user text:

Example task arguments:
<The specified parameters are not valid>

Figure 4.1- Network Traffic Alert

The same step will be repeated for every monitored resource, but each will call its related **VB** file, that will write the suitable entry in the AlertsDataset input file (the structure of this file will be demonstrated in the next section).

The complete **VB** files code exists in Appendix-A, each one of them writes its own alert value in the AlertsDataset input file when it's being called, table 4.4 below shows the name of each file, and the value that will be written in the AlertsDataset input file. The last column (Action Taken) shows the action taken of the **IPS** whenever an alarm is triggered. This will be explained in details in section 4.4-IPS Actions.

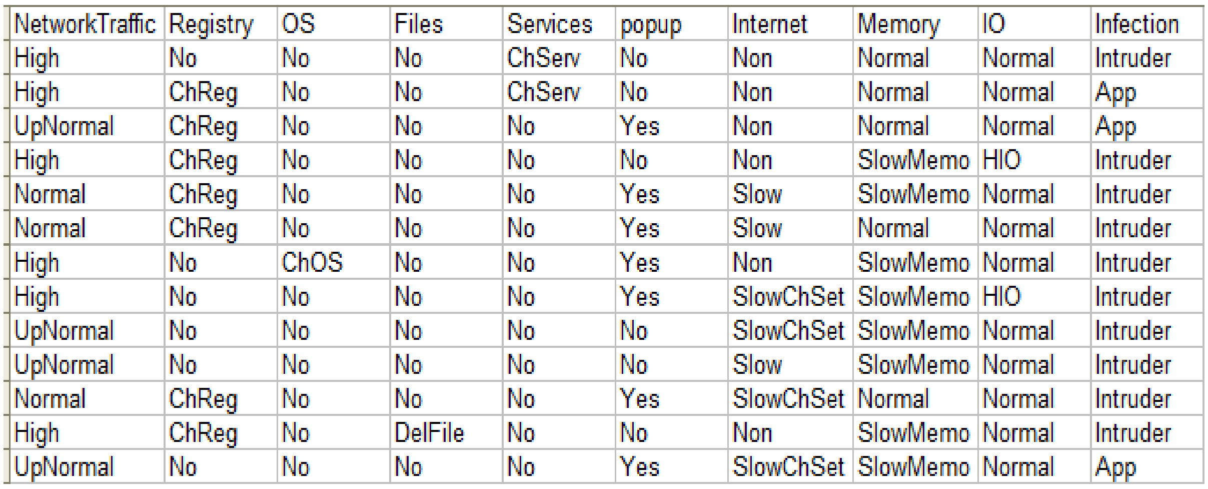
Table 4.4- VB programs written values.

Resource	VB File	Written Output Value	Action Taken
NetworkTraffic	NetworkTraffic.exe	High	Alarm the user & Reset Connection
		Normal	
		UpNormal	Alarm the user & Reset Connection
Registry	Registry.exe	ChReg	Alarm the user
		No	
OS	OS.exe	ChOS	Alarm the user and restart
		No	
Files	Files.exe	DelFile	Alarm the user and restart
		No	
Services	Services.exe	ChServ	Alarm the user and restart
		No	
popup	popup.exe	Yes	Alarm the user and restart
		No	
Internet	Internet.exe	ChSet	Alarm the user and reset connection
		Slow	
		Non	
		SlowChSet	Alarm the user and reset connection
Memory	Memory.exe	Normal	
		SlowMemo	Alarm the user & flush memory
IO	IO.exe	Normal	
		HIO	Alarm the user and flush memory
Infection	Infection.exe	App	Write Application
		Worm	Write Infection
		Trojan	

4.3- Building Alerts Dataset I/O File

When a certain monitored resource triggers an alert due to a problem, the alert in turn executes the connected **VB** file, in order to log the alarm into a dataset file named AlertsDataset, as demonstrated previously. This file is the input file to the decision tree data mining algorithm (**J48**). **J48** uses this file as an input to decide weather the logged alert indicates a real intruder action, or it is caused by a legal application.

Now it is the time to explain the structure of this file. AlertsDataset input file, is a comma separator value file. A comma separator file is too close to an excel file, consisting of columns and rows. The columns are the monitored resources mentioned in section 4.1, and the rows are the values written by **VB** alarm files, mentioned in the previous section. The figure below shows a portion of that file.



NetworkTraffic	Registry	OS	Files	Services	popup	Internet	Memory	IO	Infection
High	No	No	No	ChServ	No	Non	Normal	Normal	Intruder
High	ChReg	No	No	ChServ	No	Non	Normal	Normal	App
UpNormal	ChReg	No	No	No	Yes	Non	Normal	Normal	App
High	ChReg	No	No	No	No	Non	SlowMemo	HIO	Intruder
Normal	ChReg	No	No	No	Yes	Slow	SlowMemo	Normal	Intruder
Normal	ChReg	No	No	No	Yes	Slow	Normal	Normal	Intruder
High	No	ChOS	No	No	Yes	Non	SlowMemo	Normal	Intruder
High	No	No	No	No	Yes	SlowChSet	SlowMemo	HIO	Intruder
UpNormal	No	No	No	No	No	SlowChSet	SlowMemo	Normal	Intruder
UpNormal	No	No	No	No	No	Slow	SlowMemo	Normal	Intruder
Normal	ChReg	No	No	No	Yes	SlowChSet	Normal	Normal	Intruder
High	ChReg	No	DelFile	No	No	Non	SlowMemo	Normal	Intruder
UpNormal	No	No	No	No	Yes	SlowChSet	SlowMemo	Normal	App

Figure 4.2- AlertsDataset Input File

This file will keep growing each time a performance alarm is triggered, add to that, the last column that has the title Infection for each logged alert will not be filled by the alarm **VB** program, because this value should be decided by **J48** algorithm. This value will be an empty record (missed value) for each logged alarm, only after processing the file with **J48** algorithm, this value will be decided and then filled back, this provides a great feedback for the coming predictions to be so close to reality, and this should be increased every time the database grows. Figure 4.3 shows the decision tree output.

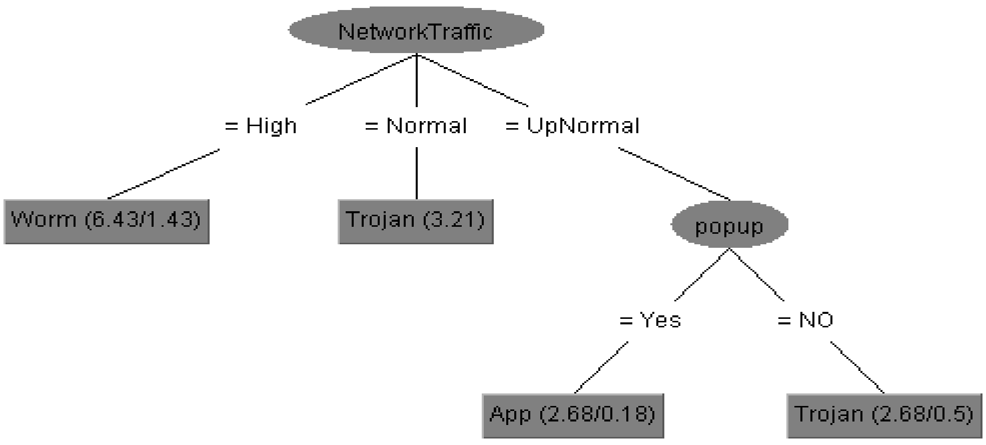


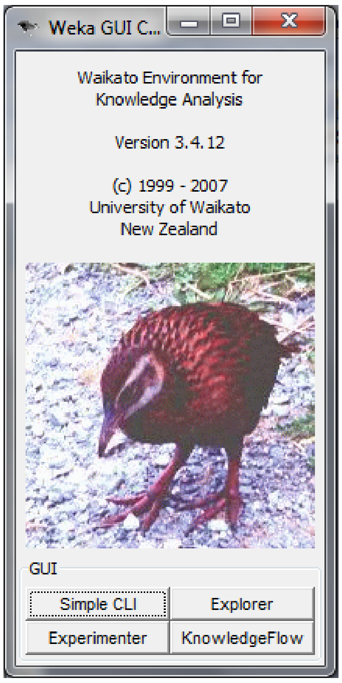
Figure 4.3- Decision Tree

4.4- Performing Decision Tree Algorithm

Now, its time to process the AlarmsDataset file with data mining techniques. Decision tree is used here to decide if an alert caused by a legal application action, or an intruder action. **J48** algorithm was chosen for such mission. J48 algorithm is the WEKA implementation of another decision tree learner named **C4.5**. **J48** was developed and refined over many years by J. Ross Quinlan of the University Sydney, Australia. Now the name **J48** is no longer a mystery, as J stands for J. Ross, and number 4 stands for the original C4.5, while number 8 stands for revision 8 of **C4.5**

Decision tree algorithms are based on a divide-and-conquer approach to the classification problem. They work from the top to down, seeking at each stage an attribute to split that best separates the classes; then recursively processing the sub problems that result from the split. This strategy generates a decision tree, which can if necessary be converted into a set of classification rules. C4.5 even has its own improvements, which are methods for dealing with numeric attributes, missing values, noisy data, and generating rules from trees.

Now it is time to load the AlertsDataset file to WEKA platform and start to analyze the outcome results. When opening WEKA application, the first graphical user interface would be as shown in figure 4.4.



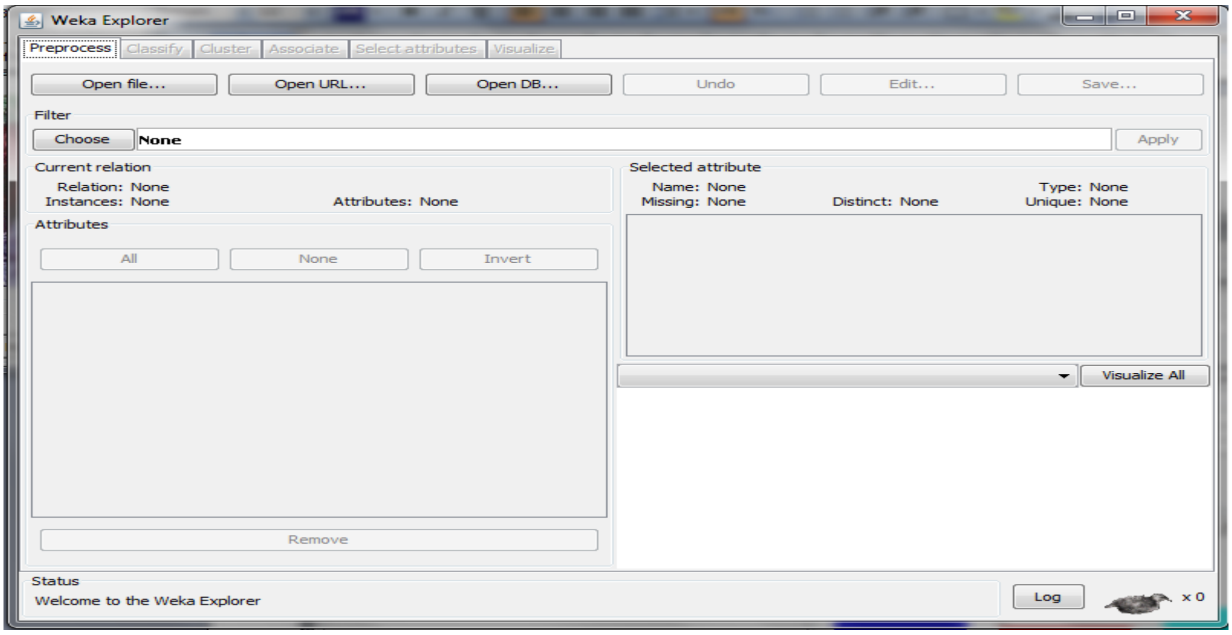
Waikato Environment for
Knowledge Analysis

Version 3.4.12

(c) 1999 - 2007
University of Waikato
New Zealand

Figure 4.4 WEKA GUI

After choosing the explorer button, another window will be opened, that looks like figure 4.5 below:



None

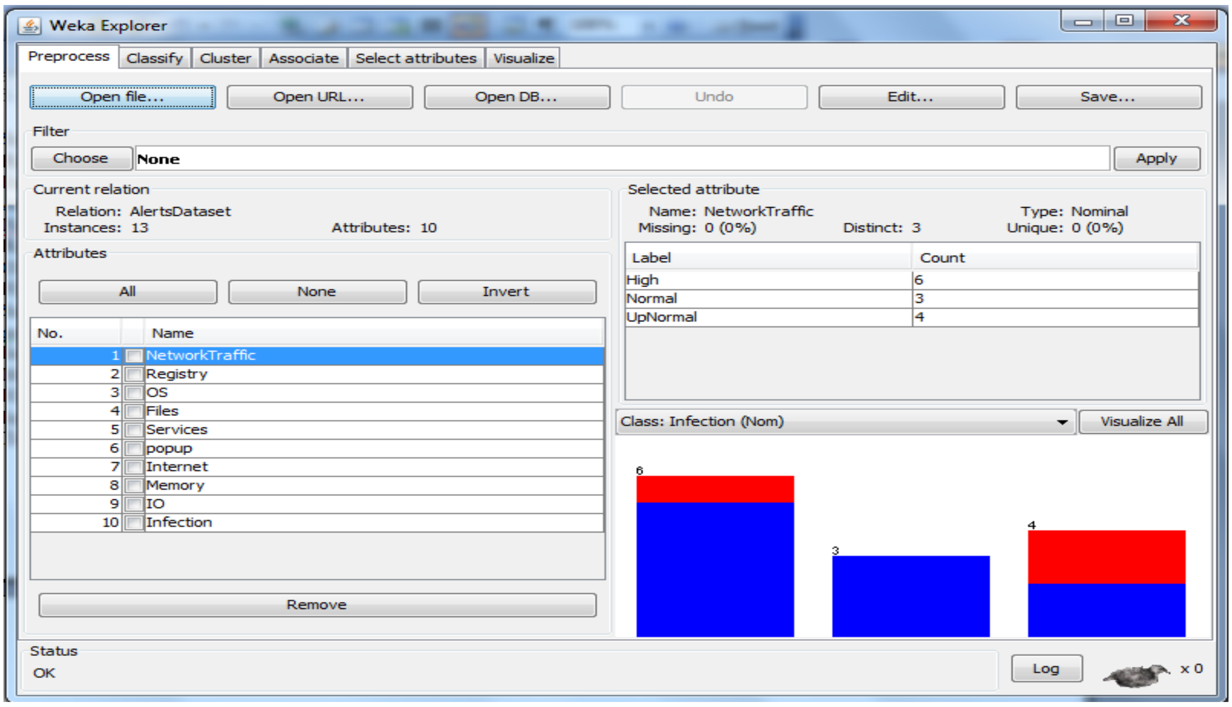
Current relation
Relation: None
Instances: None
Attributes: None

Selected attribute
Name: None
Missing: None
Distinct: None
Type: None
Unique: None

Status
Welcome to the Weka Explorer

Figure 4.5 WEKA Explore

Now it is time to hit the open file button, in order to load the AlertsDataset file, at that time the window will look like figure 4.6 below:



Filter: None

Current relation
Relation: AlertsDataset
Instances: 13 Attributes: 10

Selected attribute
Name: NetworkTraffic Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

Label	Count
High	6
Normal	3
UpNormal	4

Attributes

No.	Name
1	NetworkTraffic
2	Registry
3	OS
4	Files
5	Services
6	popup
7	Internet
8	Memory
9	IO
10	Infection

Class: Infection (Nom)

Status
OK

Figure 4.6- Open AlarmsDataset File

Notice the highlighted tags in the previous window, they show the following: relation name (AlertsDataset), number of instances (records in the file which are 13) and the number of attributes (which are 10). All attributes (10) are shown in the left pan (under the red underlined word attribute).

The columns in the right show the relation between the attribute Network Traffic and the last attribute Infection. Take the first column as an example it shows that there are six instances in the dataset, which indicate a high traffic situation, 3 of them were caused by worms, and only one was a legal application, and so on for the next two columns, but one for normal traffic and the other for the up normal traffic.

Note that the classification schemes in WEKA assume by default that the class is the last attribute in the input file.Each attribute can be visualized in the same way if it was selected from the right pan.

Now, this file should be analyzed using **J48** decision tree algorithm, to do so the classify tab must be selected, then choose button, then trees must be expanded and **J48** algorithm is selected, as shown 4.7 bellow:

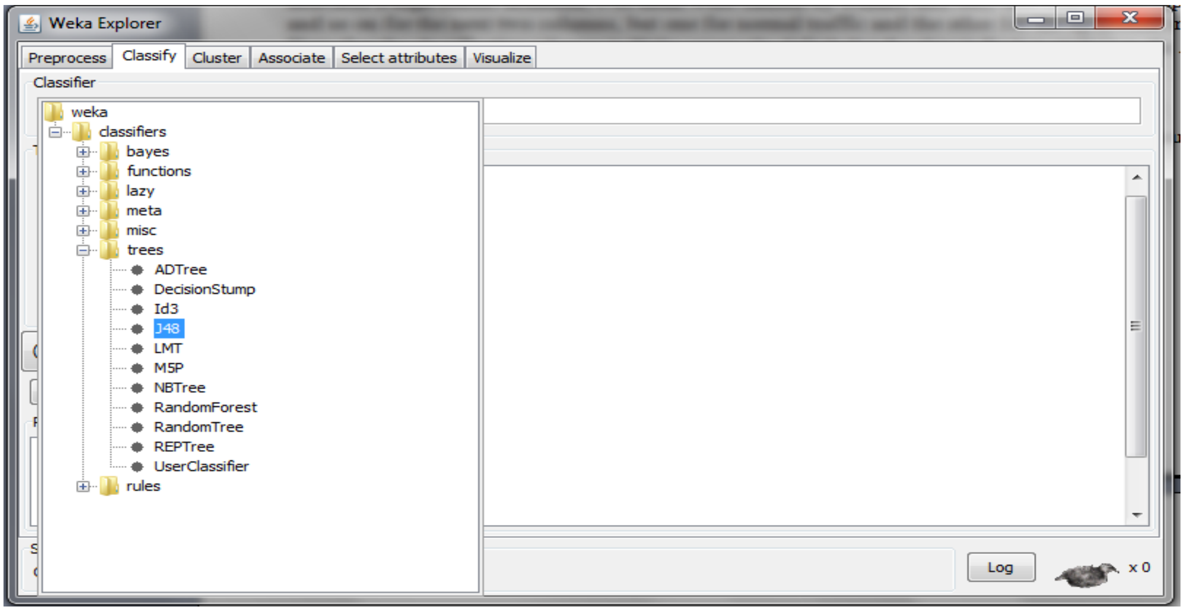
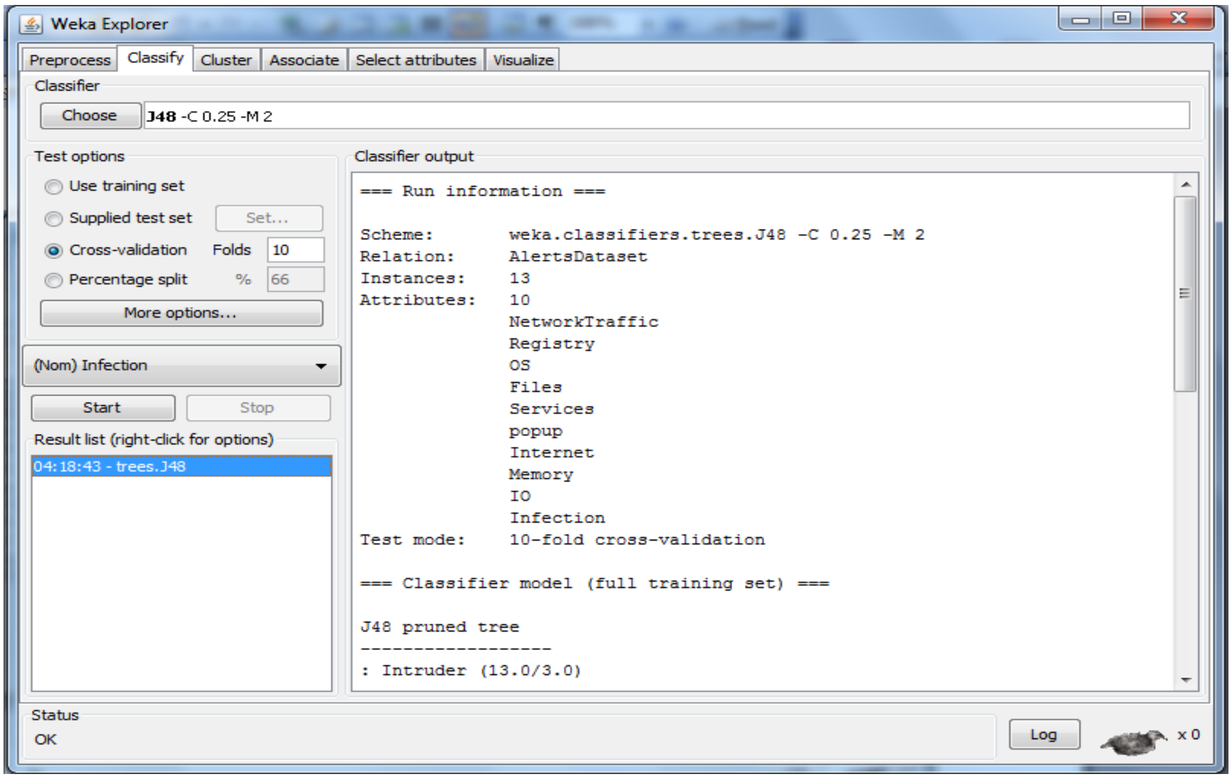


Figure 4.7- Applying J48 Algorithm

After hitting the start button, the right pan will look like figure 4.8 bellow:



Choose: J48 -C 0.25 -M 2

Test options: Cross-validation Folds 10; Percentage split % 66

(Nom) Infection

Result list: 04:18:43 - trees.J48

Status: OK

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     AlertsDataset
Instances:    13
Attributes:   10
              NetworkTraffic
              Registry
              OS
              Files
              Services
              popup
              Internet
              Memory
              IO
              Infection
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------
: Intruder (13.0/3.0)
```

Figure 4.8- J48 Algorithm Output.

This is actually the output of **J48** algorithm that will be used to expect new values, figure 4.9 bellow shows the complete output:

```
 ===Run information===

Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     PcMonitor
Instances:    31
Attributes:   9
              NetworkTraffic
              Registry
              OS
              Files
              Services
              Internet
              Memory
              IO
              Infection
Test mode:    10-fold cross-validation

 ===Classifier model (full training set=== (

J48 pruned tree
------------------
 :Int (31.0/6.0(

Number of Leaves  :   1

Size of the tree :        1

Time taken to build model: 0 seconds

 ===Stratified cross-validation===
 ===Summary===

Correctly Classified Instances          25               80.6452 %
Incorrectly Classified Instances         6               19.3548 %
Kappa statistic                          0
Mean absolute error                      0.3172
Root mean squared error                      0.402
Relative absolute error                 96.288 %
Root relative squared error              99.9966 %
Total Number of Instances               31

 ===Detailed Accuracy By Class===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
   0.893        1      0.806         1         1  Int
      0        0         0         0         0  App

 ===Confusion Matrix===

  a  b   <-- classified as
 | 0  25 a = Int
 | 0  6  b = App
```

Figure 4.9- J48 Algorithm Output

The output of the J48 algorithm of AlertsDataset is explained below:

```
 ===Run information===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     PcMonitor
Instances:    31
Attributes:   9
              NetworkTraffic
              Registry
              OS
              Files
              Services
              Internet
              Memory
              IO
              Infection
Test mode:    10-fold cross-validation
```

The first part (Run information) has the following information:

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

weka.classifiers.trees.J48: the name of the decision tree algorithm **J48**.

C=0.25 :stands for coverage of a rule, Convergance indicates the number of instances for which it predicts correctly—this is often called *its support.*

M 2: stands for minimum accuracy which is minimum number of instances that it predicts correctly.

Relation: **AlertsDataset**, the name of the relation.

Instances: 31, the number of records in the AlertsDataset.

Attributes: 10, the number of attributes in the AlertsDataset, which are:

NetworkTraffic
Registry
OS
Files
Services
popup
Internet
Memory
IO
Infection

Test mode: 10-fold cross-validation, the validation test used in the **J48** algorithm.

Error rate is used to validate the used **J48** classifier's performance; the **J48** classifier predicts the class of each instance: if it is correct, that is counted as a success; if not, it is an error. The error rate is just the proportion of errors made over a whole set of instances, and it measures the overall performance of the classifier. Now for a complete testing, the error rate should be measured in both the training dataset and a testing dataset from the same source. But for small dataset, it is used as both training and testing dataset, this happens by splitting the data into two parts and using the first one as a training set, and the other as a testing set, then do the test again but after reversing the parts of the two datasets, that's why the word cross appears.

Now, if the data set was divided into ten parts instead of two, and the same test performed on every single part as a training set while the other 9 parts as a testing set, this is called 10-fold cross validation.

```
 === ===Classifier model (full training set=== (

J48 pruned tree
------------------
 :Int (31.0/6.0(

Number of Leaves  :   1

Size of the tree :        1

Time taken to build model: 0 seconds
```

The second part shows a text description of the decision tree. It was first split depending on Int (which means intruder). In the first line, 31 instances reached that leaf, of which 6 were classified incorrectly (6 application), while the rest of the leaves have no incorrect classified instances.

```
 ===Stratified cross-validation===
 ===Summary===

Correctly Classified Instances         25               80.6452 %
Incorrectly Classified Instances        6               19.3548 %
Kappa statistic                    0
Mean absolute error                     0.3172
Root mean squared error                    0.402
Relative absolute error               96.288 %
Root relative squared error              99.9966 %
Total Number of Instances               31

 ===Detailed Accuracy By Class===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
   0.893        1     0.806         1         1  Int
      0         0         0          0         0  App
```

The next part gives an estimate of the trees predictive performance. It was obtained using cross validation of 10 folds. As it is shown about 19% of the instances (6 out of 25) has been misclassified in the cross validation, This indicates that the results obtained from the training data are optimistic compared with what might be obtained from an independent test set from the same source.

Table 4.5 shows the performance measures and their formula.

- Kappa statistic is used to measure the agreement between predicted and observed categorizations of a dataset.
- Mean-squared error is used as a performance measure.
- Mean absolute error is an average of the magnitude of the individuals.

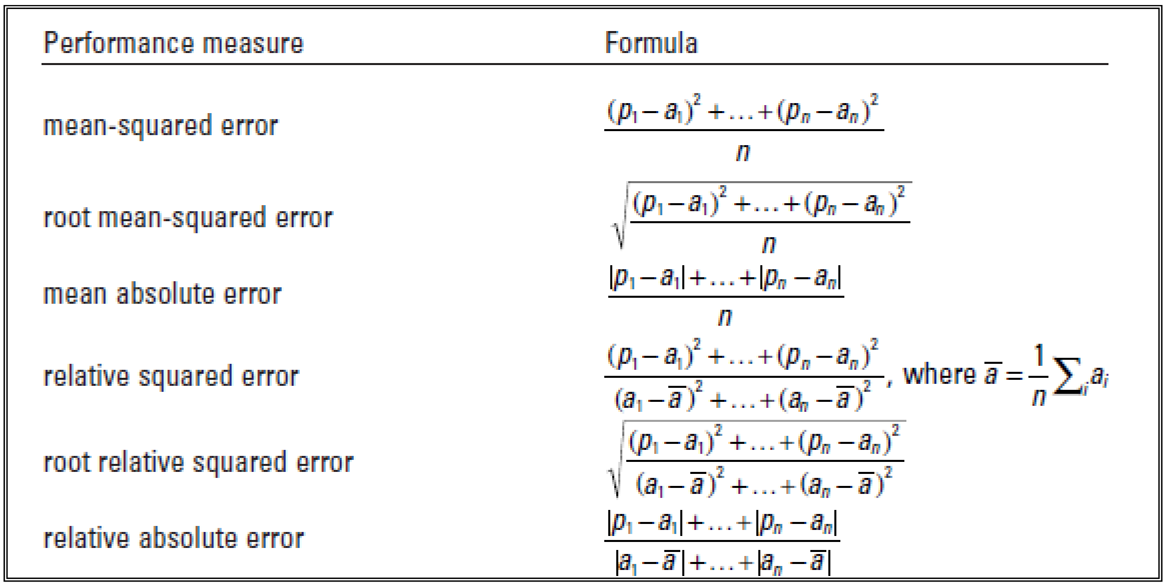
- errors without taking account of their sign.

p1, p2, . . ., pn: are the predicted values on the test instances.

a1, a2, . . ., an: are the actual values.

pi : the numeric value of the prediction for the *i*th test instance.

Table 4.5- J48 Performance Measure Formulas



Performance measure	Formula
mean-squared error	$\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}$
root mean-squared error	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}}$
mean absolute error	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{n}$
relative squared error	$\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\ldots+(a_n-\bar{a})^2}$, where $\bar{a}=\frac{1}{n}\sum_i a_i$
root relative squared error	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\ldots+(a_n-\bar{a})^2}}$
relative absolute error	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{\lvert a_1-\bar{a}\rvert+\ldots+\lvert a_n-\bar{a}\rvert}$

The true positives (TP) and true negatives (TN) are correct classifications. A false positive (FP) occurs when the outcome is incorrectly predicted as yes (or positive) when it is actually no (negative). A false negative (FN) occurs when the outcome is incorrectly predicted as negative when it is actually positive. Table 4.6 shows the combination between predicted and actual events.

Table 4.6- Prediction Verses Actual Values

		Predicted values	
		Yes	No
Actual Values	Yes	True Positive (TP)	False Negative (FN)
	No	False Positive (FP)	True Negative (TN)

Recall = number of elements retrieved that are relevant / the total number of elements that are relevant

Precision = number of elements retrieved that are relevant/ total number of elements that are retrieved

TP rate = TP/(TP+FN)

FP rate = FP/(FP+TN)

Precision = TP/(TP +FP)*100%

Recall = TP/(TP+FN)*100%

```
===Confusion Matrix===

 a  b   <-- classified as
| 0  25 a = Int
| 0  6  b = App
```

In a multiclass prediction, the result of a test set is often displayed as a two dimensional confusion matrix with a row and column for each class. Each matrix element shows the number of test examples for which the actual class is the row and the predicted class is the column.

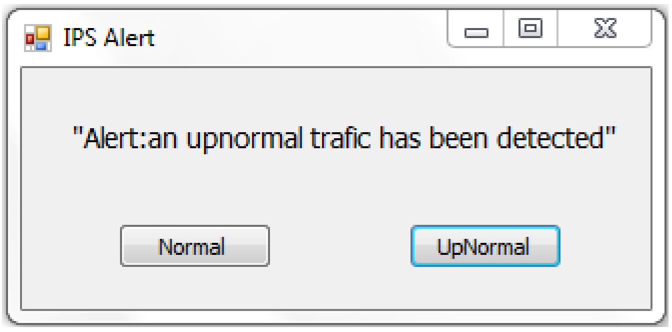
From the confusion matrix at the end it shows that none of the Int (intrusion) class were classified incorrectly, but 6 of the App (Application) instances was classified as an Intrusion.

4.5- Building IPS Actions

Referring to table 4.4 each system alert takes a certain action to prevent the intruder. This action is programmed using VB.net. Added to that, this action is also committed in the AlertsDataset file in order to reflect the actual case of the system, since the alert could be a false positive or false negative one.

For example, when a high traffic is being noticed, the user is being informed, using a simple graphical user interface window. At that window the user has the chance to commit the action as a normal one if it was caused by a legal installed new application, for example, by hitting the Normal button, or if it was a real intruder, by hitting the UpNormal button.

The button named Normal writes the entry App to the last field of the resource alert record in the AlertsDataset, while the button named UpNormal writes Intruder at the same place, to provide a feedback that finally causes elimination of false positive or false negative alerts with time. Figure 4.10 shows the Graphical user Interface for such **IPS** action.


IPS Alert

"Alert:an upnormal trafic has been detected"

Figure 4.10 Network Intrusion Alerts

The rest actions will have the same interface but the action taken will differ according to the alert recourse as previously mentioned in table 4.4, the VB.net coding is shown in the appendix for all the involved alerts.

4.6-Testing

Three test scenarios were used, in the first one the conficker virus was injected, while boot.ini and desktop.ini were injected in the second scenario, and lastly a network traffic inspection application named Net Flow analyzer was ejected to the system, the result was as follows:

Applying conficker virus:

Table 4.7 shows that, the value for Correctly Classified Instances of the AlertsDataset before applying the test was 78.125% . After applying the test the following happened:

Table 4.7-J48 output

Viruses	Triggered Alarms	Previous Correctly Classified Instances	Correctly Classified Instances	Committed Correctly Classified Instances
conficker	High Traffic & service change	78.125%	80.6452%	81.25 %

1. Two alarms on the system: High Traffic and Service Change.
2. The output for the **J48** algorithm had the following value for Correctly Classified Instances = 80.6452%

As shown in Table 4.7 the Correctly Classified Instances value is 80.6452% , which indicates that the action was classified as intrusion. Although un-triggered alarms are being represented as missing values in the AlarmDataset file.

Committing the action back to the input file increased Correctly Classified Instances value about 3.125% , this happened because the last attribute in AlertsDataset value named Infection is no longer a missing value. As it was before predicting, it was written back that this value is an infection, which caused this increase. This will help in future prediction, as the predicting rate will be more realistic, as False positive and false negative decisions will decrease with time.

Applying boot.ini / desktio.ini virus:

Table 4.8 shows that, the value for Correctly Classified Instances of the AlertsDataset before applying the test was 81. 25% . After applying the test the following happened:

Table 4.8-J48 output

Viruses	Triggered Alarms	Previous Correctly Classified Instances	Correctly Classified Instances	Committed Correctly Classified Instances
Boot.ini Desktop.ini	Slow memory	81.25%	81.25%	81.8182 %

1. Two alarms on the system: Slow memory.
2. The output for the J48 algorithm had the following value for Correctly Classified Instances = 81.25%, which indicates an infection.

Committing the action back to the input file increased Correctly Classified Instances value about 0.5682% , this happened because the last attribute in AlertsDataset value named Infection is no longer a missing value. As it was before predicting, it was written back that this value is an infection, which caused this increase. This will help in future prediction, as the predicting rate will be more realistic, as False positive and false negative decisions will decrease with time.

1. **Applying Net Flow Analyzer:**

Table 4.9 shows that, the value for Correctly Classified Inctances of the AlertsDataset before applying the test was 81. 8182% . After applying the test the following happened:

Table 4.9-J48 output

Viruses	Triggered Alarms	Previous Correctly Classified Instances	Correctly Classified Instances	Committed Correctly Classified Instances
Net Flow Analyzer	Slow memory	81.8182 %	81.8182 %	79.4118 %

1. One alarm on the system: Slow memory.
2. The output for the J48 algorithm had the following value for Correctly Classified Instances = 81.8182 %, which indicates an infection.

Committing the action back to the input file decreased Correctly Classified Instances value about 2.4064% , this happened because the last attribute in AlertsDataset value named Infection is no longer a missing value. As it was before predicting, it was written back that this value is an infection, which caused this increase. This will help in future prediction, as the predicting rate will be more realistic, as False positive and false negative decisions will decrease with time.

Chapter Five
Conclusions and Future Works

5.1- Introduction

Now and after performing all the testing, it is about the time to talk about the conclusions. But before that, the difficulties that were faced when working on this thesis are mentioned below :

1. Building the baseline that indicates each resource natural behavior was difficult, but after discovering the features that existed in Windows 7.0 admin tools, it was solved.

2. Deciding the attribute structure of the input file was critical, it had been enhanced many times before it took its final form.
3. Deciding the file format for the previous file was a major problem, because it was processed by WEKA, System alerts and VB.

There were limitations regarding the testing phase, as it was a risk to inject a virus into a system, because it could spread into the whole network. So the testing could not be performed on real servers for example

5.2- Conclusions

Using data mining in the detecting mechanism of **IPS**, has directed **IPS** to a new vision of not to depend on a database of attack signatures to inspect intruders, which eventually must be updated.

Using computer system resources as input attributes for data mining techniques provided dynamic real inputs, that were so close to reality, which caused decisions taken by **IPS** to be more objective.

Depending on such resources to be used as input values for **IPS**, simplified the diction mechanism and made it efficient at some time. This helped **IPS** to be spread on the top of each PC in the network, and maximized security level.

After applying the three test scenarios explained in the previous chapter the following conclusions could be figured:

1. Using **J48** decision tree algorithm in **IPS** detecting mechanism indicated intruders in an acceptable rate.
2. Using system resources, such as memory and network indicated anomaly acts, depending on deviations in previously measured natural baseline behavior.
3. The percentage of prediction could be tuned to reflect realistic values, by providing a feed back to the same input file through committing **IPS** actions back to the file, and analyzing it once again using the same **J48** algorithm.

4. Natural acts that trigger an alarm in system resources, due to unexpected use of such resources which happen to be above the natural baseline, are treated as an intrusion and this is called the false positive alarm. This is healthy, and could be treated in two ways: either to tune the baseline of the related resource, or depending on the feedback of the IPS that will commit the wrong alarm in the dataset, as a normal application, which causes the correctly classified instances rate to decrease, and this feedback with sufficient time will tune the system by itself.
5. As the input file grows up, the percentage of prediction will be more representative of the reality.

5.3- Future Works

1. Enlarging the number of monitored resources to cover the whole existed computer resources.
2. Centralize the management of proposed IPS by collecting the alerts log file into a single data base.
3. Enhance the feedback mechanism by assessing the triggered alerts by a specialist rather than depending on client judgment.

References

[1] Ala Al-Hamami and Saed Al-Ani, Technology of Information Security and Protection Systems, Al-Awael for Publishing and Distributing, 2007.

[2] Ala Al-Hamami and Mohammed Al-Hamami, Data mining: Concepts, Techniques and Applications, Al-Awael for Publishing and distributing, 2007.

[3] Andre' Muscat , A Log Analysis based Intrusion Detection System for the creation of a Specification Based Intrusion Prevention System, University Of Malta, 2003.

[4] Anshu Veda and Prajakata Kalekar and Anirudha Bodhankar ,Introusion Detection Using Data Mining,2001.

[5] Cumhur Doruk Bozagac ,Application of Data Mining based Malicious Code Detection Techniques for Detecting new Spyware,
Bilkent University, 2005.

[6] Daniel T. Larose, Discovering Knowledge In Data, An Introduction to Data Mining, Central Connecticut State University, Wiley Interscience, 2005.

[7] Daniel T. Larose, Data Mining Methods And models, Central Connecticut State University, Wiley-Interscience, 2006.

[8] Daniela Schiopu, Irina Tudor, Analyzing Information Security Issues Using Data Mining Techniques, University of Ploiesti, 2008

[9] Earl Carter and Jonathon Hogue, Intrusion Prevention Fundamentals, Cisco Press, 2006

[10] Harold Tipton and Micki Krause, Information Security Management Handbook, 5th edition, Auerbach Publications, 2004.

[11] Ian Witten and Eibe Frank, Data Mining Practical Machine Learning Tools and Techniques, 2nd Edition, Elsevier, 2005.

[12] Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Simon Fraser University, Morgan Kaufmann Publishers,2000.

[13] Juan Pablo Pereira, Comparison of Firewall, Intrusion Prevention and Antivirus Technologies, Juniper Networks, 2004.

[14] W.lee and S.Stolfo. Data Mining Approaches For Intrusion Detection Models, 7th USENTX Security Symposium, Vol. 7, 1998.

[15] Mahmood Hossain and Susan M. Bridges, A Framework For An Adaptive Intrusion Detection System With Data Mining, Mississippi State University, 2001.

[16] Marcos M. Campos and Boriana L. Milenova, Creation and Deployment of Data Mining-Based Intrusion Detection Systems in Oracle Database 10g, Oracle Data Mining Technologies, 2004.

[17] Mukkamala, R., Gagnon, J. And Jaiodia, S. Integrating Data Mining Techniques with Intrusion Detection Methods, International Conference on Database Security, Vol. 171, Page 33-46, 1999.

[18] Ronald L.Kurtz and Russel Dean Vines, The CISSP Prep Guide: Golden Eddition, Weily Publishing 2003.

[19] Sourour Meharouech, Adel Bouhoula, Tarek Abbes,A security policy and Network Cartography based Intrusion Detection and Prevention Systems,Higher School of Telecommunications SupCom, 2009.

[20] Theodoros Lappas and Konstantinos Pelechrinis, Data Mining Techniques for (Network) Intrusion Detection Systems, 2006.

[21] Theodoros Lappas and Konstantinos Pelechrinis, Data Mining Techniques for (Network) IntrusionDetection Systems,UC Riverside, 2007.

[22] Van Der Aalst and De Medeiros, Proccess Mining and security:Detecting Anomalous Process Executions and checking Process Conformance, University of technology Eindhoven,2005.

[23] Wenke Lee and Salvatore J. Stolfo, Data Mining Approaches for Intrusion Detection, Columbia University,2000.

[24] Zhaohui Tang and Jamie MacLennan, Data Mining with SQL Server 2005**,** Indianapolis,2005.

Appendices

Appendixes from A to C respectively, shows J48 decision tree algorithm, proposed **IPS** code written in **VB**.Net, and the complete input file in **ARFF** format.

Appendix-A

J48 Source Code In Java

```
// Generated with Weka 3.6.1
package weka.classifiers;
import weka.core.Attribute;
import weka.core.Capabilities;
import weka.core.Capabilities.Capability;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.RevisionUtils;
import weka.classifiers.Classifier;
public class WekaWrapper
  extends Classifier {
   * Returns only the toString() method.
   * @return a string describing the classifier
  public String globalInfo() {
    return toString();
   * Returns the capabilities of this classifier.
   * @return the capabilities
  public Capabilities getCapabilities() {
    weka.core.Capabilities result = new weka.core.Capabilities(this);
    result.enable(weka.core.Capabilities.Capability.NOMINAL_ATT
RIBUTES);
    result.enable(weka.core.Capabilities.Capability.BINARY_ATTRI
BUTES);
    result.enable(weka.core.Capabilities.Capability.UNARY_ATTRI
BUTES);
    result.enable(weka.core.Capabilities.Capability.EMPTY_NOMIN
AL_ATTRIBUTES);
    result.enable(weka.core.Capabilities.Capability.NUMERIC_ATT
RIBUTES);
    result.enable(weka.core.Capabilities.Capability.DATE_ATTRIBU
TES);
    result.enable(weka.core.Capabilities.Capability.MISSING_VALU
ES);
    result.enable(weka.core.Capabilities.Capability.NOMINAL_CLA
SS);
    result.enable(weka.core.Capabilities.Capability.BINARY_CLAS
S);
    result.enable(weka.core.Capabilities.Capability.MISSING_CLAS
S_VALUES);
    result.setMinimumNumberInstances(0);
    return result;
   * only checks the data against its capabilities.
   * @param i the training data
  public void buildClassifier(Instances i) throws Exception {
    // can classifier handle the data?
    getCapabilities().testWithFail(i);
   * Classifies the given instance.
   * @param i the instance to classify
   * @return the classification result
  public double classifyInstance(Instance i) throws Exception {
    Object[] s = new Object[i.numAttributes()];
    for (int j = 0; j < s.length; j++) {
      if (!i.isMissing(j)) {
        if (i.attribute(j).isNominal())
          s[j] = new String(i.stringValue(j));
        else if (i.attribute(j).isNumeric())
          s[j] = new Double(i.value(j));
    // set class value to missing
    s[i.classIndex()] = null;
    return WekaClassifier.classify(s);
```

```
   * Returns the revision string.
   * @return        the revision
  public String getRevision() {
    return RevisionUtils.extract("1.0");
   * Returns only the classnames and what classifier it is based on.
   * @return a short description
  public String toString() {
    return "Auto-generated classifier wrapper, based on
weka.classifiers.trees.J48 (generated with Weka 3.6.1).\n" +
this.getClass().getName() + "/WekaClassifier";
   * Runs the classfier from commandline.
   * @param args the commandline arguments
  public static void main(String args[]) {
    runClassifier(new WekaWrapper(), args);}
class WekaClassifier {
  public static double classify(Object[] i)
    throws Exception {
    double p = Double.NaN;
    p = WekaClassifier.N19fc4ac0(i);
    return p;
  static double N19fc4ac0(Object []i)
    double p = Double.NaN;
    if (i[3] == null) {
      p = 0;
    } else if (((Double) i[3]).doubleValue() <= 0.6) {
```

```
p = 0;
 } else if (((Double) i[3]).doubleValue() > 0.6) {
 p = WekaClassifier.N292e601(i);
 return p;
static double N292e601(Object []i) {
 double p = Double.NaN;
 if (i[3] == null) {
   p = 1;
 } else if (((Double) i[3]).doubleValue() <= 1.7) {
 p = WekaClassifier.N1a3de412(i);
 } else if (((Double) i[3]).doubleValue() > 1.7) {
   p = 2;
 return p;
static double N1a3de412(Object []i) {
 double p = Double.NaN;
 if (i[2] == null) {
   p = 1;
 } else if (((Double) i[2]).doubleValue() <= 4.9) {
   p = 1;
 } else if (((Double) i[2]).doubleValue() > 4.9) {
 p = WekaClassifier.N154aa5a3(i);
 return p;
static double N154aa5a3(Object []i) {
 double p = Double.NaN;
 if (i[3] == null) {
   p = 2;
 } else if (((Double) i[3]).doubleValue() <= 1.5) {
   p = 2;
 } else if (((Double) i[3]).doubleValue() > 1.5) {
   p = 1;
 }     return p;
```

Appendix-B
IPS Code

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop


Public Class Form1

    Private WekaOutput As Integer

    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        If TextBox1.Text = "" Then
            MessageBox.Show("Enter the number of ")
        Else
            Dim i As Integer
            Dim ExcelFile As Excel.Application
            Dim xlAlertsDataset As Excel.Workbook
            Dim xlWorkSheet As Excel.Worksheet
            Dim xlWorkSheetRowID As Excel.Worksheet

            ExcelFile = New Excel.ApplicationClass
            xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
            xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
            xlWorkSheetRowID =
xlAlertsDataset.Worksheets("sheet2")
            WekaOutput = TextBox1.Text

            For i = 2 To
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1))

                If System.Convert.ToString(xlWorkSheet.Cells(i, 1)) =
"High" Then '''NetworkTraffic
                    If WekaOutput >= 50 Then
                        If MessageBox.Show("Network Traffic is Hight, if
this is normal please click on No", "Network Traffic",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                            xlWorkSheet.Cells(i, 10) = "Introder"
                        Else
                            xlWorkSheet.Cells(i, 10) = "App"
                    End If
                    xlWorkSheet.Cells(i, 11) = "Alarm the user & Reset
Connection"
                  End If
                End If

                If System.Convert.ToString(xlWorkSheet.Cells(i, 1)) =
"UpNormal" Then """NetworkTraffic
                  If WekaOutput < 50 Then
                    If MessageBox.Show("Network Traffic is Up
Normal, if this is normal please click on No", "Network Traffic",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                      xlWorkSheet.Cells(i, 10) = "Introder"
                    Else
                      xlWorkSheet.Cells(i, 10) = "App"
                    End If
                    xlWorkSheet.Cells(i, 11) = "Alarm the user & Reset
Connection"
                  End If
                End If

                If System.Convert.ToString(xlWorkSheet.Cells(i, 2)) =
"ChReg" Then """Registry
                  If WekaOutput >= 50 Then
                    If MessageBox.Show("There is somthing wrong on
registry, if this is normal please click on No", "",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                      xlWorkSheet.Cells(i, 10) = "Introder"
                    Else
                      xlWorkSheet.Cells(i, 10) = "App"
                    End If
                    xlWorkSheet.Cells(i, 11) = "Alarm the user"
                  End If
                End If

                If System.Convert.ToString(xlWorkSheet.Cells(i, 3)) =
"ChOS" Then  """OS
                  If WekaOutput >= 50 Then
                    If MessageBox.Show("There is somthing wrong on
OS, if this is normal please click on No", "OS alarm",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                        xlWorkSheet.Cells(i, 10) = "Introder"
                    Else
                        xlWorkSheet.Cells(i, 10) = "App"
                    End If
                    xlWorkSheet.Cells(i, 11) = "Alarm the user and
restart"
                End If
            End If

            If System.Convert.ToString(xlWorkSheet.Cells(i, 4)) =
"DelFile" Then """Files
                If WekaOutput >= 50 Then
                    If MessageBox.Show("There is somthing wrong on
system Files, if this is normal please click on No", "File Alarm",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                        xlWorkSheet.Cells(i, 10) = "Introder"
                    Else
                        xlWorkSheet.Cells(i, 10) = "App"
                    End If
                    xlWorkSheet.Cells(i, 11) = "Alarm the user and
restart"
                End If
            End If

            If System.Convert.ToString(xlWorkSheet.Cells(i, 5)) =
"ChServ" Then """Services
                If WekaOutput >= 50 Then
                    If MessageBox.Show("There is somthing wrong on
system services, if this is normal please click on No", "services",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                        xlWorkSheet.Cells(i, 10) = "Introder"
                    Else
                        xlWorkSheet.Cells(i, 10) = "App"
                    End If
                    xlWorkSheet.Cells(i, 11) = "Alarm the user and
restart"
                End If
            End If
```

```
                If System.Convert.ToString(xlWorkSheet.Cells(i, 6)) =
"Yes" Then """popup
                    If WekaOutput >= 50 Then
                        If MessageBox.Show("There is somthing wrong on
popup, if this is normal please click on No", "popup",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                            xlWorkSheet.Cells(i, 10) = "Introder"
                        Else
                            xlWorkSheet.Cells(i, 10) = "App"
                        End If
                        xlWorkSheet.Cells(i, 11) = "Alarm the user and
restart"
                    End If
                End If

                If System.Convert.ToString(xlWorkSheet.Cells(i, 7)) =
"ChSet" Then """Internet
                    If WekaOutput >= 50 Then
                        If MessageBox.Show("There is somthing wrong on
Internet, if this is normal please click on No", " Internet",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                            xlWorkSheet.Cells(i, 10) = "Introder"
                        Else
                            xlWorkSheet.Cells(i, 10) = "App"
                        End If
                        xlWorkSheet.Cells(i, 11) = "Alarm the user and
reset connection"
                    End If
                End If

                If System.Convert.ToString(xlWorkSheet.Cells(i, 7)) =
"SlowChSet" Then """Internet
                    If WekaOutput >= 50 Then
                        If MessageBox.Show("There is somthing wrong on
Internet, if this is normal please click on No", "Internet",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                            xlWorkSheet.Cells(i, 10) = "Introder"
                        Else
                            xlWorkSheet.Cells(i, 10) = "App"
                        End If
                        xlWorkSheet.Cells(i, 11) = "Alarm the user and
reset connection"
                    End If
                End If

                If System.Convert.ToString(xlWorkSheet.Cells(i, 8)) =
"SlowMemo" Then """Memory
                    If WekaOutput >= 50 Then
                        If MessageBox.Show("There is somthing wrong on
Memory, if this is normal please click on No", "Memory",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                            xlWorkSheet.Cells(i, 10) = "Introder"
                        Else
                            xlWorkSheet.Cells(i, 10) = "App"
                        End If
                        xlWorkSheet.Cells(i, 11) = "Alarm the user & flush
memory"
                    End If
                End If

                If System.Convert.ToString(xlWorkSheet.Cells(i, 9)) =
"HIO" Then """IO
                    If WekaOutput >= 50 Then
                        If MessageBox.Show("There is somthing wrong on
IO, if this is normal please click on No", "IO alarm",
MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes
Then
                            xlWorkSheet.Cells(i, 10) = "Introder"
                        Else
                            xlWorkSheet.Cells(i, 10) = "App"
                        End If
                        xlWorkSheet.Cells(i, 11) = "Alarm the user and
flush memory"
                    End If
                End If

            Next

            xlAlertsDataset.Save()
            xlAlertsDataset.Close()
            ExcelFile.Quit()
        End If
```

```
    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

        'Try
        '    Dim ExcelFile As Excel.Application
        '    Dim xlAlertsDataset As Excel.Workbook
        '    Dim xlWorkSheet As Excel.Worksheet

        '    ExcelFile = New Excel.ApplicationClass
        '    xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\output.xls")
        '    xlWorkSheet = xlAlertsDataset.Worksheets("output")
        '
MessageBox.Show(System.Convert.ToString(xlWorkSheet.Cells(1
0, 14)))
        'Catch ex As Exception
        '    MessageBox.Show(ex.Message) """("Please the check the
CSV file", "Error")
        'End Try

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs)

    End Sub

    Private Sub Label1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Label1.Click

    End Sub
End Class
```

Alerts Code

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop

Module Module1

    Sub Main()
```

```
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1

        xlWorkSheet.Cells(ERowID, 10) = "App"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()
    End Sub

End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop
Module Module1

    Sub Main()
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1
```

```
        xlWorkSheet.Cells(ERowID, 10) = "Worm"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()

    End Sub

End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop

Module Module1

    Sub Main()
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1

        xlWorkSheet.Cells(ERowID, 10) = "Trojan"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()
    End Sub

End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop
Module Module1

    Sub Main()
```

```
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1

        xlWorkSheet.Cells(ERowID, 7) = "ChSet"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()



    End Sub

End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop

Module Module1

    Sub Main()
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")
        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1

        xlWorkSheet.Cells(ERowID, 9) = "HIO"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()
    End Sub

End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop

Module Module1

    Sub Main()
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1

        xlWorkSheet.Cells(ERowID, 8) = "SlowMemo"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()
    End Sub

End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop
```

```
Module Module1

    Sub Main()

        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1

        xlWorkSheet.Cells(ERowID, 1) = "High"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()


        'releaseObject(ExcelFile)
        'releaseObject(xlAlertsDataset)
        'releaseObject(xlWorkSheet)
    End Sub

    Private Sub releaseObject(ByVal obj As Object)
        Try

System.Runtime.InteropServices.Marshal.ReleaseComObject(obj)
            obj = Nothing
        Catch ex As Exception
            obj = Nothing
        Finally
            GC.Collect()
        End Try
    End Sub
End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop

Module Module1

    Sub Main()
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1

        xlWorkSheet.Cells(ERowID, 1) = "UpNormal"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()
    End Sub

End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop

Module Module1

    Sub Main()
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1

        xlWorkSheet.Cells(ERowID, 3) = "ChOS"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()
    End Sub

End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop

Module Module1

    Sub Main()
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1
        xlWorkSheetRowID.Cells(1, 1).value = ERowID

        xlWorkSheet.Cells(ERowID, 2) = "ChReg"

        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()
    End Sub

End Module
```

```
Imports Microsoft.Office
Imports Microsoft.Office.Interop

Module Module1

    Sub Main()
        Dim ExcelFile As Excel.Application
        Dim xlAlertsDataset As Excel.Workbook
        Dim xlWorkSheet As Excel.Worksheet
        Dim xlWorkSheetRowID As Excel.Worksheet

        ExcelFile = New Excel.ApplicationClass
        xlAlertsDataset =
ExcelFile.Workbooks.Open("C:\FilesAlearts\AlertsDataset.xls")
        xlWorkSheet = xlAlertsDataset.Worksheets("sheet1")
        xlWorkSheetRowID = xlAlertsDataset.Worksheets("sheet2")

        Dim ERowID As Integer
        ERowID =
System.Convert.ToInt16(xlWorkSheetRowID.Cells(1, 1).value) + 1

        xlWorkSheet.Cells(ERowID, 5) = "ChServ"
        xlWorkSheetRowID.Cells(1, 1).value = ERowID
        xlAlertsDataset.Save()
        xlAlertsDataset.Close()
        ExcelFile.Quit()

    End Sub
```

Appendix-C

Input ARFF File

```
@relation PcMonitor_predicted

@attribute Instance_number numeric
@attribute NetworkTraffic {Yes,No}
@attribute Registry {ChReg,NO}
@attribute OS {ChOS,NO}
@attribute Files {DelFile,NO}
@attribute Services {ChServ,NO}
@attribute Internet {Slow,ChSet,SlowChSet,Non}
@attribute Memory {SlowMemo,Normal}
@attribute IO {HIO,Normal}
@attribute predictedInfection {Int,App}
@attribute Infection {Int,App}

@data
0,Yes,ChReg,NO,NO,NO,Slow,Normal,Normal,Int,App
1,No,NO,NO,NO,NO,SlowChSet,SlowMemo,Normal,Int,Int
2,Yes,NO,NO,DelFile,NO,Non,SlowMemo,Normal,Int,Int
3,Yes,ChReg,NO,DelFile,NO,Non,SlowMemo,Normal,Int,Int
4,Yes,ChReg,NO,NO,NO,Non,Normal,Normal,Int,App
5,No,ChReg,NO,NO,NO,Slow,SlowMemo,Normal,Int,Int
6,No,NO,NO,NO,NO,Non,SlowMemo,HIO,Int,Int
7,No,NO,NO,NO,NO,SlowChSet,SlowMemo,Normal,Int,App
8,No,ChReg,NO,NO,NO,Slow,Normal,Normal,Int,Int
9,Yes,NO,NO,NO,NO,Slow,SlowMemo,Normal,Int,Int
10,No,NO,NO,NO,NO,SlowChSet,SlowMemo,Normal,Int,App
11,Yes,ChReg,ChOS,DelFile,ChServ,SlowChSet,SlowMemo,Nor
mal,Int,Int
12,No,NO,NO,NO,NO,SlowChSet,SlowMemo,Normal,Int,Int
13,Yes,NO,NO,NO,NO,SlowChSet,SlowMemo,Normal,Int,App
14,Yes,ChReg,NO,NO,ChServ,Non,Normal,Normal,Int,Int
15,Yes,NO,NO,NO,NO,SlowChSet,SlowMemo,HIO,Int,Int
16,Yes,ChReg,ChOS,DelFile,ChServ,SlowChSet,SlowMemo,Nor
mal,Int,App
17,Yes,ChReg,NO,DelFile,NO,Non,SlowMemo,Normal,Int,Int
18,Yes,NO,ChOS,NO,NO,Non,SlowMemo,Normal,Int,Int
19,Yes,ChReg,NO,NO,ChServ,Non,Normal,Normal,Int,Int
20,Yes,ChReg,ChOS,NO,NO,Non,SlowMemo,Normal,Int,Int
21,Yes,NO,ChOS,NO,NO,Non,SlowMemo,Normal,Int,Int
```

```
22,Yes,NO,NO,NO,ChServ,Non,Normal,Normal,Int,Int
23,No,NO,NO,NO,NO,SlowChSet,SlowMemo,HIO,Int,Int
24,No,ChReg,NO,NO,NO,Slow,SlowMemo,Normal,Int,Int
25,No,ChReg,NO,NO,NO,SlowChSet,Normal,Normal,Int,Int
26,Yes,NO,NO,NO,NO,Slow,SlowMemo,Normal,Int,Int
27,No,ChReg,NO,NO,NO,Slow,Normal,Normal,Int,Int
28,Yes,NO,NO,NO,NO,SlowChSet,SlowMemo,Normal,Int,Int
29,Yes,ChReg,NO,NO,NO,Non,SlowMemo,Normal,Int,Int
30,No,ChReg,NO,NO,NO,SlowChSet,Normal,Normal,Int,Int
```