OXFORD BROOKES UNIVERSITY

School of Technology

Module P00999

Dissertation

Semester 2 – 2005

"The Future of Intrusion Detection & Prevention"

Submitted by : NACMIAS Illan 04064270

Supervisor : BASU Kashinath

I. ABSTRACT

I have been involved in the computer security field for many years, and I have decided to use this dissertation to apply ideas that I have for reducing the problem of network intrusion. As you know, for several years, the world of computing has been suffering from the same plague: misdemeanour on the network system. We will assess what it is the financial loss due to these misdemeanours, to underline the importance of figuring out this problem. However, the main goal of this dissertation is to survey the current technologies to restrict this plague and what techniques we shall use in the forthcoming years. Throughout this survey, we will underline many important points, as the flaws of the actual Intrusion Detection System, and how to reduce the impact of these flaws on our system. Besides, within this paper, we will try to apply methods from my taught part in **Oxford Brookes University** like, *Bayesian Inference*, *Artificial Intelligence* techniques, *and Autonomous agents*. The key ideas are to put reasoning in the correlation of our log files, in order to automate daily tasks, also to make our alerts more reliable and accurate. About the correlation, we advise using *Data Mining* techniques; the application of Data Mining to intrusion detection issue dates from few years. In addition we will highlight what Data Mining algorithm is most suitable for detecting intrusion. We will base this investigation on the implementation of a log analyser program which is able to classify log content into two groups “normal behaviour of the system” and “abnormal behaviour” then we will study the false positive and negative rate of our implementation in order to highlight which Classification algorithm is the most suited.

Likewise, in the new technologies used for intrusion detection, there is the analogy between human-immune system and computer security, we will reuse some of the techniques present in the immune system, because the goal of the immune system is the same as our goal (detecting and removing all malicious and suspicious events in our system).

The challenge of this dissertation is not to revolutionize the network detection and prevention, but to highlight relevant methods in order to suggest to an analyst new tools, new algorithms, and new techniques surely most suited to its problems. To meet this challenge, we will provide an intrusion detection architecture able to detect and to respond to any intrusion attempt and likewise a well-defined model based on Data Mining for intrusion detection.

Keywords: Intrusion Detection System, Intrusion Prevention System, false positive, false negative, Data Correlation, Alert Correlation, Data Fusion, Data Mining, human immune system, Autonomous intelligen agent, Bayesian inference.

II. ACKNOWLEDGEMENT

I wish to thank several persons for their support during the long writing of my dissertation. First of all I give thanks to my tutor for his relevant advice and his availability, to Phil Torr (especially for his lecture about Bayesian reasoning) and to all teachers from the module P00485 (Artificial Intelligence), likewise I thank Stephanie Forrest from the University of New Mexico for all the documents she made available about the analogy between human-immune system and computer security. I also want to thank all Oxford Brookes University staff for their kindness.

Furthermore, I also thank my family, my grandparents, my dear parents (Yossef & Dorit) and my amazing brother Yarden. Finally I wish to thank, lots of friends and relatives for their help all along this work : Pierre Murasso, Steve Fouhal, Eli & Freidy Brackman, Gad Schaffer, Daniel White, Avichag, Esteban & Léo, Annie Nomat, Lucien Ginzburg and Naomi Shemer for her sweet melodies.

TABLE OF CONTENTS

TABLE OF FIGURES

III. INTRODUCTION

Nowadays the problem for the computing security industry is to provide enough safety for the customers, providers and companies. In 2004 we have in the mean for the 1st semester 10,6 attacks per day and 13,6 attacks per day for the last semester and when we know that **the cost of eight days of massive worm attacks (Win.32 Blaster) in August 2003 may be up to $2 billion** (in clean up expenses and lost productivity), we can easily assert that the financial loss due to network incidents is close to **$ 5-10 billions per year**, which represents the entire budget of a small country such as Slovenia.

Furthermore, from now on not only the big companies are attacked, also the small firms are vulnerable to hackers, fraudsters, theft or industrial espionage. Besides, according to Jeremy Beale, the author of the famous virus Blaster, "*the collective apathy of many is creating the ideal environment for the spread of digital crime*". Apart from that, it is unlikely that the attackers will stop using the Internet and abusing the vulnerabilities of widespread software at one stroke. Nevertheless, we increase the repressive measures against the attackers. That's why in 2004, 82% of the companies are using a security policy or semi-skilled tools and this percentage is growing over the years according to Csi/Fbi. Administrators know that this plague will not stop suddenly. Thus, the government and specialized organizations need to make a more intensive advertising about the "eCrime", to highlight the lack of awareness of "*many*". In addition, security companies need to provide better products to contain attacks and above all better methods of detection.

Moreover, we have to find a way for helping the analyst in his day job, so we can highlight how much effective-work time he loses in order to make up for an infection of 10% of 5500 vulnerabilities listed in 2004:

> An analyst needs to read the descriptions of the vulnerabilities (around **20min to read a description**), it takes **1 hour to install the correct patch**, so the **loss of time productivity will be 69 days** ((5500*10%) * 1hour) and the loss can be higher if the administrator reads all the security news and descriptions and patches a single system (loss of 298 days). So, over one year a perfectly efficient administrator will spend almost all his work time repairing vulnerabilities.

This Dissertation is an overall picture of the current and future techniques in Intrusion Detection. There is a real need to prevent the intrusion attempts on our network and to be able to respond to any intrusion, within the first chapter we will sum up the criticals flaws of the current Intrusion Detection system, then we will highlight new methods like Data Correlation, Alert Fusion, Data Mining, analogy between human-immune system and computer security, in order to prevent the weaknesses of the systems. Following that, we will provide a survey of different Data Mining algorithms in order to underline which is most suitable for Intrusion Detection. We will base this survey on the development of a Log analyser providing to us relevant statistics results.

The main goal of this dissertation is to provide an Intrusion Detection guidebook for the security analysts, to make them understand that the problem of Intrusion Detection is surely the biggest one in the world of computing, despite the fact that we have enough knowledge and techniques to prevent this problem. The other goal of this dissertation is to survey Data Mining algorithms in order to highlight the most suitable method for intrusion detection. Finally, the personal goal of this dissertation is to use this work to assert that I am capable of working as a computer security consultant.

IV. INTRUSION & PREVENTION DETECTION SYSTEM

A. IDS & IPS

Within a security survey of *Ernst & Young* in 1999, US department of defence did network penetration tests, and highlighted that only 4% of the systems broken were able to detect the attack, this fact implies that we need a system for detecting network intrusion.

So, to the question "What is an Intrusion Detection System (IDS)?" it is surprising to note that we get, as much different answers as persons to whom we have asked the question. Thus, in order to give an appropriate definition of the assignment of IDS, we quote the definition of Thierry Evengelista (the chief project of the IDS Dragon, and also consultant in security for Network Associates). He defines an IDSs like that: "***An IDS is a set of softwares or hardwares, having for main goal the detection, the analyse of all attempts to distort the information system***". An IDS is most likely likened to a burglar alarm, something that gives information about past and ongoing activity, to facilitate the detection of suspicious and possibly wrongful activity. Thus, it can detect event that may or may not be an intrusion. In addition, it makes further use of firewall, because firewall is comparable to locked doors. Their uses are complementary, as the use of IPSs with IDSs and firewalls. IDSs work at the network layer of the OSI model (*this layer translates logical network addresses and names to their physical addresses and is responsible for addressing and managing network problems such as packet switching, data congestion, and routing*).

IDSs sensors are positioned at choke points on the network to analyse packets and detect a possible known malicious sequence. Thus, IDSs are similar to antivirus software in that they use known signatures to recognize traffic patterns that maybe malicious, and then they alert the analyst.

Fig 1. The main process for IDS is that a NIDS or HIDS passively gathers information and pre-processes and classifies them. Analysing an engine determines whether the information falls outside normal activity, and if so, it is then matched against a knowledge base. If a match is found, an alert is sent. The Alert manager communicates to the analyst an appropriate response to the threat.

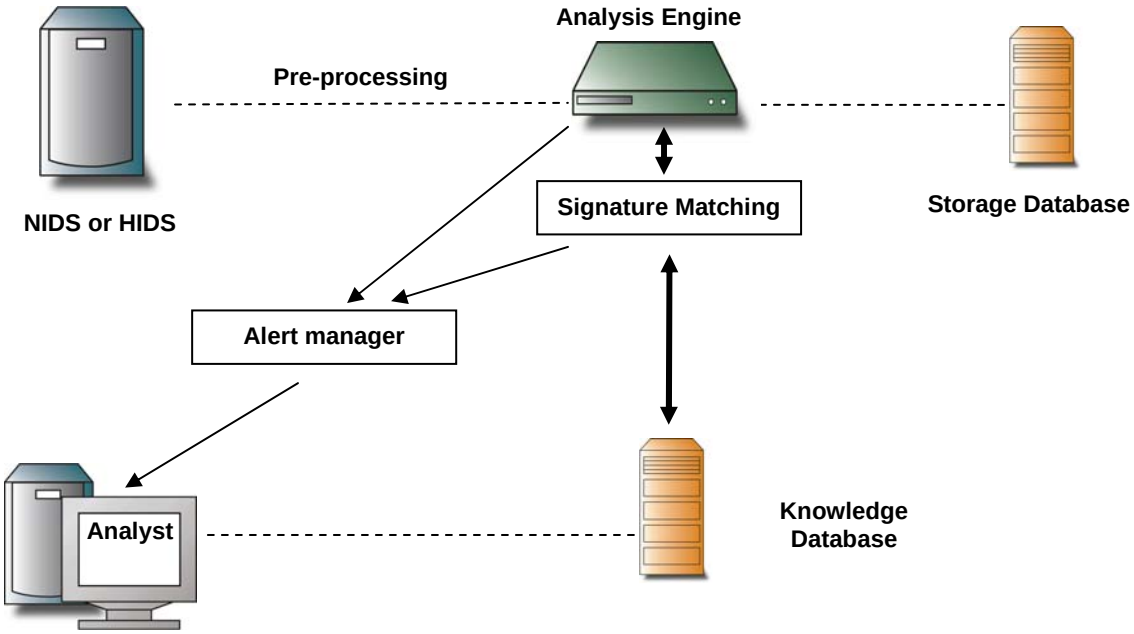

Figure 1 : Standard IDS system (source: Intrusion Detection & Prevention 2004 by Carl Endorf, Eugene Schultz and Jim Mellander)

There are two important categories of IDSs: **NIDS**s and **HIDS**s. We will quickly describe their gear.

- **NIDS** (Network Intrusion Detection System): Sensor who sounds and analyses in inactive way the flow of data, that are currently passing through the network and detect the intrusions in real-time.

These sensors are spread in the main choke points of the network. NIDS system of detection is based on **Pattern-matching** technique (Identify attacks, by scanning the traffic and determining the signatures of known attack patterns). So, the system catches only known attacks, if the attacks-list is constantly updated. Unfortunately, attackers, can easily evade to the NIDSs, they know exactly how, the NIDS system work, so they just need to modify a known attack (make slight alteration in the attack) in order to elude the monitoring of the NIDS.

- **HIDS** (Host Intrusion Detection System): Agent put into a computer, it is analysing, protecting the logs and the outgoing and coming flow of information. So the HIDS sensors examine the log activities files, in order to find signatures of known attacks.

NIDS and HIDS are genuinely related and complementary, however HIDS has an advantage over the NIDS, because its sensors are directly spread on a targeted-server, so in the case of ciphered log or files, the sensors can easily get the deciphered data, because it is on the end of the communication network. So the coded files are decoded by the target and then spread on the network, and HIDS sensors can catch them.

The other important tool is not doing network detection, but network prevention. This tool is called **IPS** (Intrusion Prevention System) and also used **HIPS** and **NIPS**. IPSs sit online and are designed to be defensive measures, that stop the negative consequences of attacks on networks like guard dogs. IPSs are inspecting all packets going inbound or outbound. IPS monitors the network and can initiate a response or send an alarm to the analyst when it detects a suspicious traffic. So, if an attack is already list, the IPS will certainly prevent it.

This is unlike IDSs, which is applied after the attack intrusion. Distinctness with IDSs is that, IDSs sit on network and are more passive, because the review of the attack is making after the misdemeanour. Furthermore IDSs are better at detecting hacking attacks, whereas IPSs are better for blocking web defacement. In addition, whereas a NIDSs simply detect suspicious traffic, IPSs suggest a prevention response to the administrator of the system. That reduces the risk of suffering damage and the financial loss due to intrusions.

Figure 2 Sensors deriving from the source of information will interpret the network traffic and will perform packet analysis. The traffic is then forward to the **detection engine**. During the scanning, it is built a reference table that classifies the information and helps the **traffic shaper** manage the flow of the information. The **detection engine** does pattern matching against the known attacks database. Finally, the **Alert manager** communicates to the analyst a suitable response to the threat.

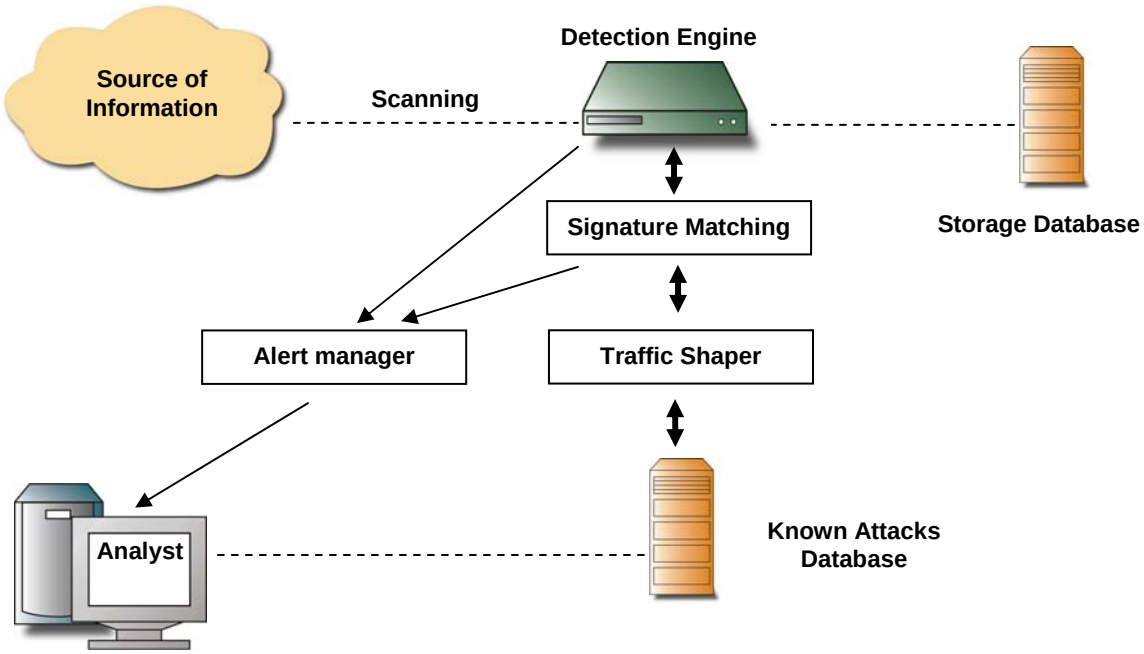

Figure 2 : Standard IPS system (source: Intrusion Detection & Prevention 2004 by Carl Endorf, Eugene Schultz and Jim Mellander)

Thus, currently IDS/IPS are the best tradeoffs in order to detect and prevent a network intrusion. According to Csi/Fbi (2004) 68% of the organizations using security tools are using IDS, and 44% an IPS, those percentages have been increasing since 2 years ago, so we can plan that IDS/IPS will soon be widespread like anti-virus and firewalls. Although IDS/IPS seem to be the perfect tools to block and to prevent a misdemeanour, there are flaws and problems with their use and setting-up. One of the major problems is that we cannot identify an attempt to intrude with a reliability of 100%. So, for the moment we survey the different techniques of detection used into IDS/IPS and then we highlight what are the actual flaws of IDS/IPS.

B. TECHNIQUES OF DETECTION

In this part, we will classify all the network detection methods. We can divide them into two different approaches: **Misuse detection** & **Anomaly detection**.

Misuse Detection: How to detect an attempt to intrude according to the current behaviour of the user and independently of its past actions.

Most of the techniques originally from the Misuse Detection approach are based on the detection of suspicious signatures. “Intrusion scenario” can be used instead of Misuse detection to mean a description of a fairly precisely known kind of intrusion. A system based on Misuse detection continually compares the current system activity to a set of intrusion scenarios in order to detect intrusion attemp in progress.

As stated by Carl Endorf, Jim Mellander and Eugene Schultz in their book “*Intrusion Detection & Prevention*” (2004) Let’s look at the four main phases of the analysis process of a Misuse Detection system:

- **Pre-processing** The first step is to collect data about intrusions, vulnerabilities, and attacks, and put them into a Classification scheme or pattern descriptor. From the Classification scheme, a behavioural model is built, and then put into a common format.
- **Analysis** The event data are formatted and compared with the knowledge base by using a pattern-matching analysis engine. This engine looks for defined patterns that are known as attacks.
- **Response** If the event matches the pattern of an attack, the engine then sends an alert. If the event is a biased match, the next event is examined. Note that biased matches can only be analyzed with a stateful detector, which has the ability to maintain state. Different responses can be returned depending on the specific event records.
- **Accuracy** of pattern-matching analysis comes down to updating signatures, because an IDS is only as good as its latest signature update (like antivirus). This is one of the flaws of pattern-matching analysis. However, most IDSs allow automatic and manual updating of attack signatures.

Now let's break down into the three different methods of network detection originally from Misuse Detection:

- **Pattern-matching :** Identify attacks, by scanning the traffic and determining the signatures of known attack patterns
- **Protocol analysis:** Check the accordance of the traffic regarding the RFCs. Likewise, watching suspect parameters similar to some protocols, in order to make sure that these will not be used in a malicious way by an attacker.
- **Heuristic method:** Close to the virus detection method of the anti-virus systems.

Anomaly Detection: Contrary to the Misuse Detection approach, the Anomaly Detection approach consists of detecting an intrusion attempt according to the past behaviour of a user, noticing that the ultimate goal is to draw up a well-defined user profile, regarding its working customs. That's why Anomaly Detection is also referred to as **profile-based detection**.

The different methods of network detection originally from Anomaly Detection will be analysed in the next chapter. Now let's review the analysis process of this approach of detection, still according to Carl Endorf, Jim Mellander and Eugene Schultz.

- **Pre-processing** The first step in the analysis process is collecting the data in which behaviour considered normal on the network is baselined over a period of time. The data are put into a numeric form and then formatted. Then the information is classified into a statistical profile that is based on different algorithms in the knowledge base.
- **Analysis** The event data are typically reduced to a profile vector, which is then compared to the knowledge base. The contents of the profile vector are compared to a historical record for that particular user, and any data that fall outside of the baseline normal activity is labelled a deviation.
- **Response** At this point, a response can be triggered either automatically or manually.
- **Accuracy** The data records must be kept updated. The profile vector history will typically be deleted after a specific number of days. In addition, different weighting systems can be used to add more weight to recent behaviours than past behaviours.

Figure 3 sums up the main network detection methods. In the aftermath of this survey, it is obvious that we will not focus on the method already commercialised, i.e. not on the Misuse Detection techniques. In the upcoming chapters, we will develop a brand new

method employing new techniques originally from the Anomaly Detection approach and new ideas deriving from Artificial Intelligence.

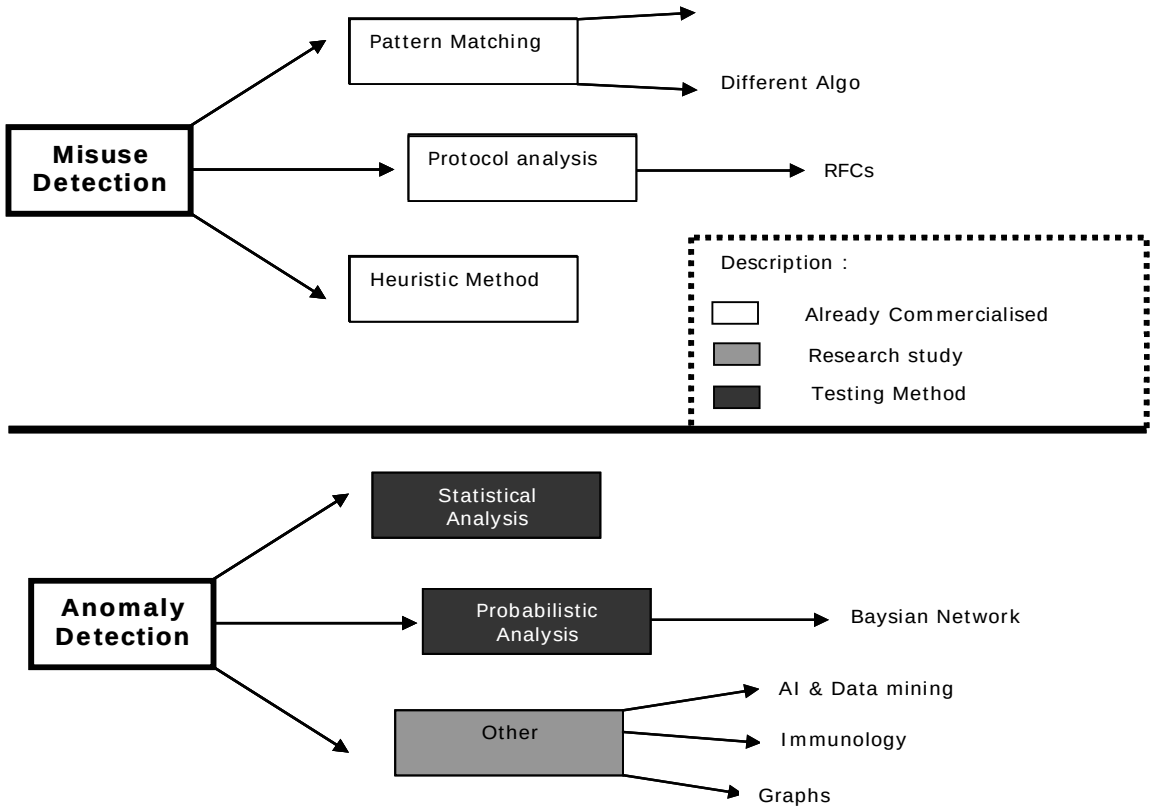

Figure 3 : Summary of all network detection techniques

C. WEAKNESSES OF IDS/IPS

In this sub-section, we will survey the technical boundaries of IDS/IPS. These boundaries can be due to network abuse, complexity of configuration and administration, chronic problems etc. Noticing that, Intrusion detection technology is still growing and improving whereas Intrusion Prevention technology is in its infancy, that's why there are several critical flaws in IDS/IPS.

Now we are going to highlight the main weaknesses of IDS/IPS and especially the false positive and negative, which are becoming the main issue in Intrusion Detection.

- **False positive and negative**: When something occurs that causes the system to incorrectly identify an intrusion when none has occurred (false positive), or to incorrectly fail to identify an intrusion when one has in fact has occurred (false negative). Thus, the problem is to know, how to parameter IDS/IPS in an effective way, in order to avoid all the pretending-alerts. Furthermore, the intrusion detection system can be made more sensitive at the risk of introducing false positive, or it can be more restrictive, at the risk of rejecting true positive. Therefore, there are tradeoffs between, false positive and false negative, because, if we attempt to restrict false positive, we can favour the expansion of false negative and vice versa. Moreover, the false positive notion is relatively independent of the detection method used. For example: A system based on the Pattern-matching method will generate as many false alerts as a system based on Protocol analysis. Noticing that Pattern-matching generates false positives because a pattern is badly interpreted, whereas the Protocol analysis risks generating alerts, simply because the use of an option is made in a none specified way by a RFC. Besides, according to a survey of Marcus J.Ranum for ICSA Labs in 2003, the rate of **false alerts per IDSs/IPSs log file can reach 10%**. Thus, we really need to reduce this huge amount of false alerts, that's why within the next chapters, we will discuss about new methods resolving this problem. Besides, we can measure the rate of false positive and negative by the following relations :

$$\textit{False positive rate} = \frac{\textit{False positive}}{\textit{Classified as an Intrusion + False positive}}$$

$$\textit{False negative rate} = \frac{\textit{False negative}}{\textit{Classified as normal + False negative}}$$

True positive is the number of correct classified network attacks, whereas True negative is the number of correct data non classified as network attacks.

- **How identify accurately an intrusion**: IDSs detect all the modification in the system, even though the modification is due to a normal use of the system and not from an internal or external network-based attack. So, we have to find an accurate way to define what an intrusion is, in order not to confuse it with the analysis of log files.

- **Limits due to High Bandwidth:** The firsts IDSs were built 8 years ago, and the bandwidth was very weak at this time, thus IDSs processing time were adapted to this bandwidth. Henceforth, IDSs are not able to treat packages, without loss, because the speed of IDSs processing doesn't fit the bandwidth of the network. We thus understand why the IDSs processing becomes critical and why we need to have the best IDS system, if it is only able to analyse one packet of two. This problem increases the slow reactivity of the IDSs, and it becomes almost impossible to be aware of an attempt to intrude in time.

- **The standardization (lack of interoperability):** There is a real lack of interoperability (*the ability to exchange and use information* **source:** www.answers.com). We cannot mix several technologies resulting from a different supplier. Generally, the only possibility that we have is to export log files in a standard format. For example: It is impossible to build a system using the *network sensors Enterasys*, the *host sensors Entercept* from **Network associates** and Snort IDS. That's why we need an organization similar to the **W3C** for web-development, in order to fulfil the needs of the analysts and to facilitate their work.

- **Configuration of IDS/IPS overwhelming the time of the analyst**: First of all, we need a specialist to administer our IDS/IPS configuration ($75, 000 per year), and even with a specialist, the network IDS/IPS administration can overwhelm his daytime, notably with the administration of the attacks storage base, checking how well defined the parameters of the IDS/IPS are, deployment of the sensors, patch the vulnerabilities, read the document about

the vulnerabilities exploits, analysing the alerts. So, the only solution to this problem is to automate day tasks and simply task (see chapter VII).

- **Produce too low a "return on security investment (ROSI)"**: A IDS/IPS ROSI can be understood by analysing the difference between annual loss expectancy (ALE, *it is the annually expected financial loss to an asset resulting from one threat*) without IDS/IPS and the annual loss expectancy with IDS/IPS deployment.

  The Traditional Return on Security Investment (ROSI) equation is:

  ROSI = R – ALE, where ALE = (R-E) + T

  And where R = the Annual Recovery Cost from Intrusion without IDS/IPS, E = the Annual Cost Savings gained by stopping intrusion with IDS/IPS and T = Annual Cost of IDS/IPS technology management.

  (Source "Security still a 'Net lifestyle issue" by site M.H. (Mac) McMillan, April 8th, 2002, Reprinted from tech.first, a publication of Business First.).

So, the last 3 years the ROSI of IDS/IPS was negative, partly due to the problem of false positive and negative and also the huge cost of its administration.

Once we have understood that our security tools and security policies have weaknesses, we thus understand why our security is imperfect. As in the physical world, we can achieve better security with additional resources, enhancement of the tools, better understanding of the needs, a better design and evidently more investment (especially when we know that only 1-5% of IT companies budget is expended for security reasons) . So, now the question is, are we able to achieve better security than the one we have currently?

Our response must be yes, and we will see within the forthcoming chapters, how we are capable of providing better security. First of all, we will analyse a new method, which filters out the false alerts and then we will put forward relevant architectures for network detection.

V.DATA CORRELATION

In the last chapter, we established that IDS/IPS are the best tradeoffs to detect and prevent network intrusions, although we noticed that this new system has still several serious flaws. The new method that security analysts will use in order to reduce the flaws of IDS/IPS, is called ***Data Correlation***. This brand new method provides us with more information about the analysing output of incidents, such as the degree of threat of an attack, whether events are related (data about attacks and potential attacks). So, now we will focus on how we can correlate data, and what we will obtain; besides this, we will also look at statistical correlation in order to analyse the relationship between events, then we will look at Bayesian inference.

A. BASIC APPROACH OF DATA CORRELATION

First of all, it is useful to give an accurate definition of what Data Correlation is, according to Arian Mavriqi from Graz University of Technology (2002):

"***Data correlation means associating sets of events detected through various means and applying knowledge to determine whether they are related, and if so, in what manner and to what degree. This kind of correlation requires comparing observations based on different types of parameters, such as source and destination IP addresses, an identifiable network route, commands entered by a suspected attacker, and the time when activity began or ended. Data sources can be intrusion-detection sensors, logs, databases (such as threat databases), and so forth.***"

So, regarding that definition, we need a lot of information to be able to detect correlated events into them. To amass as much data as possible, we have to use a method called: **Data Aggregation**:"(which) *refers to the process of acquiring more and more data*." According E.Waltz (1990) "*Adequately track and capture each desired type of event and data*". Therefore, we need to collect as many activity logs as possible. One new relevant method widely used for gathering logs is to set-up **honeypots**. In his book about IDS & IPS, Eugene Schultz defines a honeypot as "***a decoy server that looks and acts like a normal server, but that does not run or support normal server functions. The main purpose of deploying honeypots is to observe the behaviour of attackers in a safe environment, one in which there is no threat to normal, operational systems*** ". This tool is really improving detection and is useful for studying how an attacker is trying to trespass in our network.

However, we can widen the use of honeypot, in mixing it with data-mining methods (finding patterns in large data set). In order to learn more about honeypot, you can find in Wheatly Library rm C313, the dissertation of Melanie Woodward (2002) about "*Construction of a honeypot to monitor unauthorised systems access and the role of a honeypot as a part of a business security solution*".

Another, method is important in the Correlation process, this method is called: **Data Fusion**.

The fusion is the capacity for a system to post events obtained by diverse sources and classified by trigger sequence. Most of the time, this term is used in military circles to describe the correlation of data in time, and the assignment of importance is weighted to the output. For example: Let us suppose, that a Firewall, an antivirus and an IDS are simultaneously detecting an alert. With the use of Data Fusion, we can supply to the analyst the alert containing all the previous events (alerts) deriving from the different sources. Data Fusion reduces the amount of alerts posted considering the enormous amount of detecting events.

Despite that, the greatest barrier for using Data Fusion has been the lack of a common format for data for intrusion detection systems. The other problem is that Data Fusion, needs to be automated in our system, because the analyst does not have enough time to do the tasks of Data Fusion system manually, so in order to solve this problem, we need to use an automated intelligent agent (we will explain this technique in the next chapter).

Figure 4. summarizes all the different steps during the Data-Correlation process, we need to gather the information from the logs, then we can aggregate them, filter out the chronic alerts and finally we can correlate our data, in order to find a common pattern in our amount of information.

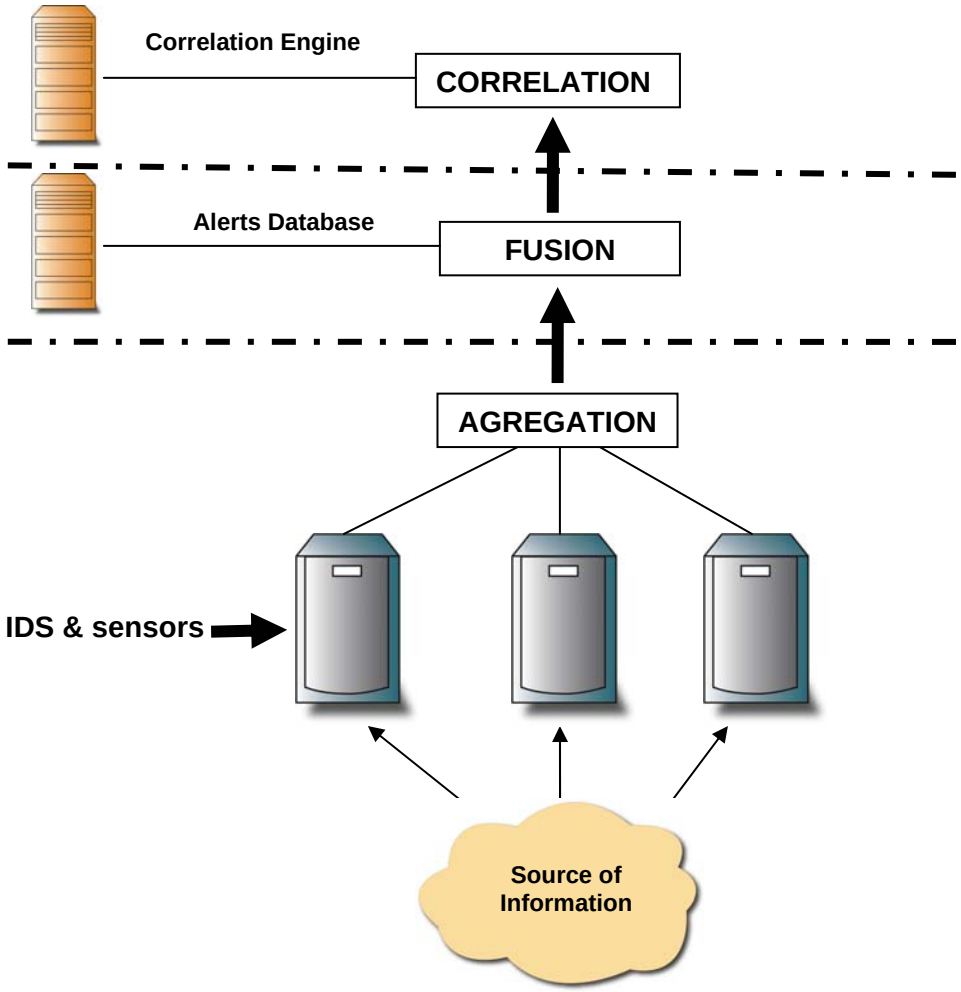

Figure 4 : Different levels of Correlation

We can also underline fours main methods of Data Correlation:

- Correlating events between attributes of similar attacks: Likewise called Alert Correlation or Alert Fusion, we will breaking down this method in Chapter VII.
- Correlating events based on predefined attack scenarios: Not relevant enough, because we don't know yet how to formulate the scenarios in an explicit way.
- Correlating events by statistical analysis: We will see in depth, how it works in the following sub-section.
- Correlating events by information of pre- and post conditions of individual attacks: Based on Bayesian inference, we also survey this method in the following sub-section.

Thus, Data Correlation is becoming an important tool for monitoring security events, as its main goal is to detect intrusion and then notify it to the analyst for a possible response. Apart from that, the ultimate goal of Data Correlation is to reduce the exposure time (the time an organization is susceptible to be attacked), and according to Winn Schwartau(1999), ***the exposure time has to be equal to the detection time + the reaction time*** (the time it takes to eliminate or resolve a problem).

Therefore, in the use of Data Correlation, we need a database with all the information about the past incidents and the activity log; it can also be interesting to mix this use with the use of a Honeypot which has already collected relevant information about the behaviour of the attackers. Then we can apply specific knowledge to our collected information to see whether events are related. We can likewise develop new knowledge methods about types of attacks and then try to prevent likely incidents.

For example, if one attacker is trying to exploit a vulnerability of our system and then break into it, with the activity log we can easily notice that there is a huge amount of network traffic from the same external network address (the one of the attackers), so the administrator can then deduce that it is likely that an attacker is attempting an attack on our system. The deduction of the administrator can be true or not (false positive), thus we need to computerize this form of deduction through a software which checks the activity log all day long, as well as the database and the honeypot to detect similar deduction with the aid of reasoning methods. In addition, to be almost sure that, we are under attack, we have to use statistical and probability methods in order to make our hypothesis more accurate.

Apart from that, Data Correlation can determine the origin, the degree of threat of an attack, so Data Correlation can piece data together in order to find out what event exactly happened, and why. This method is called **Event Reconstruction** and it is really useful for analysts and administrators, because they can thus have a better understanding of the needs of the system. We can identify what are the weaknesses of the system and then apply a better security policy to prevent the abuse of these weaknesses by attackers.

Moreover, Data Correlation can help us with the problem of “false positive” and “false negative” because we can now correlate data taking into account, in what context we are, and in what context we are using an application.

For example: If we taking our last example again, and instead of an attempt of exploit vulnerability by an attacker, we are using a Ftp, or peer-to-peer application, and we are demarcating from the logs a great amount of network traffic from the same host, then we will consider this is an attack, while it is just likely to be the result of the normal use of our peer-to-peer software.

Thus, with a usual IDS, this event would be listed in the log as an attempt to intrude whereas it is just a normal use of the system. So Data Correlation provides us with more flexibility in our day by day software use.

B. WHAT DATAS ARE CORRELATED?

At first sight, it seems obvious that we can correlate all data that we can collect (data from IDS, Firewall, Honeypot…). However, some of the data are useless to correlate, because some of them are too expensive to correlate or need too much time and sometimes overwhelm the processing of our system. So we are going to highlight what are the different advantages and disadvantages of correlating some data as opposed to others.

Thus we base our survey on information collected by **The National Centre For Supercomputing Applications (NCSA)** who have put an activity log on every applications mostly used by administrators. (For example : An Ftp log will track login of the user, Ip address of the user, Ftp commands, uploaded and downloaded files, filenames, password, rootkit, number of connections, time of connection, specific port or not.)

Thus, according to NCSA the logs mostly used in organizations are:

- Netflow / Firewall: A size of 500mo per days and contains the number of connections, the amount of bytes transferred, port used, source and destination Ip address. Provide an overall picture of the traffic (log all the traffic activity).
- DNS: Logs the requests from the clients and their answers.
- DHCP: contains Ip & MAC address and the event ID.
- HTTP: a size of few hundred of Mo per day and contains the date & time, the Ips source/destination, the event ID, and the different request and answers.
- FTP : size of tens of Mo per day, logs, login of the user, Ip address of the user, Ftp commands, uploaded and downloaded files, filenames, password, rootkit, number of connections, time of connection, specific port or not.
- SMTP: size of tens of Mo per day, logs, the email addresses of the sender, the receiver, date and time, and size of the mail.
- SSH: size of tens of Mo per day, logs, the date/time, the Ip source/destination, event ID and user' commands.
- Telnet: contains users' commands and user login/logouts.
- IDS/IPS output : a size of several hundred of Mo per day, contains date/time, source/destination Ip/port, application ID, Host ID, Event source location/destination, alarm level, code executed, executed commands, optional event.

- Database of attacks: Containing data about attacks, for example: In the event of Ip spoofing attack, the system will log (the number of new sessions opened from a source domain to the server and the number of existing sessions opened).

From the content of those logs, we can select useful information for our correlation and for a common log file format:

- **Date/Time**
- **Protocol**
- **Ip source**
- **Ip destination**
- **Port source**
- **Port destination**
- **Size of the packet**

The advantage of these logs is that most of the time, they can provide a relevant analysis of the network traffic and likewise are readily understable. However, sometimes logs can overwhelm the traffic, can be long to process and also the output quality is really variable. Moreover, we also need money to make investment in tools like Database of attacks, and the cost of setting up and administration tools can be really high.
Likewise, we can have other sources of data to correlate, such as the output of audit of vulnerability scanning, or VPN logs, routers and switch logs, honeypot logs. In addition we can use intelligent sensors to feed us in relevant data to correlate; we will develop this use in the next chapter.

Besides, we need a method for correlating all of those logs (for example: what element of Ftp log can be correlated to Database attack), which will be developed in the next sub-section.

C. WHAT KNOWLEDGE TECHNIQUES CAN WE APPLY TO DETERMINE CORRELATED-EVENTS?

Our challenge is to provide methods to express the correlation between data and reasoning about them. According to the definition of Data Correlation that we gave, we need to apply knowledge to our set of events to determine correlated-events. But what knowledge do we have to use?

We can highlight likelihood with the **Semantic-Web** field, where we need to apply knowledge on web content, in order to have more relevant, reliable information and make web content reasoning by itself. So with the aid of knowledge we have to lead an automated reasoning. Thus, we have to focus on knowledge techniques like:

- Data-mining : The principle of using Data Mining is to determine useful patterns in the data, and then use them to compute and classify what we have analysed. We will develop this topic in chapter VII.
- Rules of thumb / Heuristic Knowledge: Guide the reasoning process, empirical knowledge from experts derived from past experience.
- Fuzzy Logic: "*Model approach to correlation that can dynamically adapt to changing environment*". Use to represent ambiguity.
- Expert system: Identify indications of known attacks, can detect intrusions by implementing known intrusions scenarios. System based on a *knowledge base* and a *rule base* for guiding the reasoning. Reliable and flexible.
- Neural Networks: "*Proposed to identify characteristics of system users and identify significant variations from the user's established behaviour*". Principle advantage, able to analyse the data, even if they are incomplete or distorted.
- Semantic network and frames: Graphical representation of knowledge can be useful, for representing the relations between logs and their inheritance. Also useful for seeing is believing and avoiding distorting data.

Artificial-Intelligence researchers have already studied such systems and they proved the relevance of applying those techniques for Data Correlation. Thus, with this set of techniques, we are sure to well-correlate all the data that we have aggregated. In chapter VIII, we will focus on Data-mining, because this method helps us to highlight uncommon relations and above all, it was never implemented for Data Correlation softwares. The other techniques, especially Heuristic knowledge, and Fuzzy Logic are currently the only techniques used in the few known Data Correlation softwares (*Truthreat, risk correlation, Snort*...).

D. STATISTICAL CORRELATION

Statistical Correlations is a widely **Statistical Inference** (*inference about a population from a random sample drawn from it or, more generally, about a random process from its observed behaviour during a finite period of time*. Source: http://www.wikipedia.org) method, used to calculate the degree of threat of correlated events. The principle of this technique is really unsophisticated, we need to measure some variables (symbol for a quantity that may take on different values) and then determine the statistical report between those variables. We can for example correlate information between the number of vulnerability scans on a host, and the number of FTP-related attacks on the same host, for a time of one month, and then get a relevant result. Statistical correlation is likewise useful to provide a prediction about a correlation, for example, if we already know the correlation of 2 variables and we also know one of the variables for a given period of time, then we are able to predict the other variable (it can be the number of attacks against a server, the attempts to intrude etc).

Furthermore, one of the advantages of this technique is that we don't need any pre-existing information about attacks. However, this method may be used to detect threats on pre-defined activity thresholds (most of time defined with the survey of past experiences).

The output of such analysis is a range between -1.00 and +1.00; a negative result means that a variable increases with the other, and the inverse for a positive result, evidently a value of 0 means that there is no correlation between both variables.

E. BAYESIAN INFERENCE

This is another Statistical Inference method that can be applied to intrusion detection and use probabilities such as degrees of belief. This method allows the following of a scenario in which causality is important, and where the knowledge of all the relations for the correlation is incomplete, that's why it is necessary to describe them with probabilities.

For example: a user is requesting Brookes webpage:

- There is a connection on Brookes server :
  - **Source.4188 > www.brookes.ac.uk.80 : S 247 : 247 (0) win 64240 (DF)**
- Then, there is a high probability (maybe 90%) that the request of connection is followed by :
  - **GET http://www.brookes.ac.uk/HTTP/1.0**
- Then, we can estimate that this reply, will be followed by (with a probability of 70%)
  - **HTTP/1.1 200 ok**

Then an alert will be sent to the analyst, when the probability rate, will have exceeded a defined-threshold (for example 15%), and all information about the threshold and probabilities can be defined, by monitoring the habits of a user.

Apart from that in network detection, Bayesian inference can provide us with the probability whether a host is compromised or not, and this feature is not undertaken by current IDS/IPS, Bayesian inference is also useful for reducing false alerts. Now we will explain how the Bayes theorem works, and then we will give some relevant examples of its use in network detection.

Therefore in order to explain what this theorem consist in, we have to introduce some new terms:

- Prior probabilities: Is simply how likely we think that a hypothesis is true prior to taking into account the new evidence. ***Pr(H)***, ***Pr(¬H)***

- Likelihoods: Likelihoods are conditional probabilities, they are the probability that a given piece of evidence would obtain, given that a certain hypothesis is true. ***Pr(E/H), Pr(E/¬H)***

- Posterior probability: is the new probability that we assign to our hypothesis ***H*** after considering the new piece of evidence ***E***. There will always only be one of these, regardless of how many prior probabilities we are considering. This is what we are looking to solve for in using Bayes Theorem (***Pr(H/E)***)

Bayes Theorem:

$$Pr(H/E) = \frac{Pr(H)Pr(E/H)}{Pr(H)Pr(E/H) + Pr(\neg H)Pr(E/\neg H)}$$

(source: Philip Torr from Brookes computer vision group)

Bayesian Inference allows us to prevent intrusion by predicting and correlating events. Due to the probability output, we can decide to allow a connection or on the contrary stop it. For example: H is the event than a host has been compromised, E is a remote connection with a bad node (a compromised host which is attempting to connect to other hosts). Furthermore, we have, the probability that a host is compromised and has open connections with a compromised host: 0.7 = Pr (E/H), Pr (H) =0.2, Pr (¬H) =0.8, thus we get:

$$Pr(H/E) = \frac{0.2 * 0.7}{0.2 * 0.7 + 0.8*0.3} = 0,36$$

So, we have 36% that a host connected to a bad node has been compromised.

In this chapter, we have highlighted the importance of Data Correlation. We have listed, the different forms of data to correlate, and what knowledge to apply to them to find the relevant relationship between events for detecting intrusions. Finally, we examined techniques, which interpret the result of correlation and advise us about the likely response of prevention. Although these techniques are useful for us, they need to be automated; likewise we need to find a way to alert the analyst about a possible intrusion as soon as it has been detected. We have to remember that our principle goal is to reduce the exposure time of an attack and to be able to detect and reply in time. Thus, we will focus on all of these important points in the next part, notably with Data Fusion and Alert Correlation.

VI. INTELLIGENT INTRUSION DETECTION

In this chapter, we will underline new relevant and reliable techniques to make our intrusion detection reasoning more intelligent and to remove chronic problems of IDS/IPS like the false positive, the time that analysts need to analyse data, normalization. First of all, we will discuss about the advantages mentioned in an interesting article, having as its main goal the reduction of the amount of alerts (it will be a gain of productivity for the analyst who often is overwhelmed by the task of analysing alerts). Then, we will highlight that the use of automated agent in order to complete usual network detection tasks. And finally, we will discuss about the analogy between the human immune system and computer security.

A. ALERT FUSION

This part is deriving from an interesting article published in 2001 by Alfonso Valdes and Keith Skinner from **SRI International** called "*Probabilistic Alert Correlation*". The aim of their article is to find a relevant way for reducing the gigantic amount of alerts and among them the false positives that an analyst has to analyse every day. This is an important issue, because, as you already know, this amount of alerts is likely to overwhelm the intrusion detection analyst.

So, they have developed an **Alert Correlation** model, based on the purpose of fusing alerts, which means, *just sending a single alert instead of multiple ones whether similar intrusion detection related-events occur*. The purpose of fusion is to combine similar alerts and not to correlate closely related attempt of intrusions.

A new alert is compared to a list of existing alerts, and this alert is then correlated with its most similar alert (called meta-alert), assuming the likelihood is above a specified match threshold. Furthermore, this model is based on Bayesian inference and the similarity result is a number between 0 and 1 (similar to a Bayes probability result), computed following this equation:

$$\mathbf{SIM(X,Y)} = \frac{\sum E_j SIM(X_j, Y_j)}{\sum E_j}$$

With $\boldsymbol{X}$ = Candidate meta-alert for matching, $\boldsymbol{Y}$ = new alert, $\boldsymbol{E_j}$ = Expectation of similarity for feature j, $_j$ =Index over the alert features.

In addition, the decision to fuse alerts is based on the temporal difference between these alerts and the information that they contain. Generally, just two alerts are fused, but more than two alerts might be fused as well. We can melt alerts, only when their start-times and end-times differ by less than a certain, pre-defined time. *Figure 5*, is an example of the log output of an alert database (doing alert fusion), in this example we can fuse the different attacks, because the attacks periods are closed.

ID	Name	Sensor	Start/End	Source	Target	Tag
3	ARP flooding	S_4	9.15/11.10	81.67.45.12	Localhost	correlated
17	ARP flooding	S_1	8.40/10.15	81.67.45.12	Localhost	correlated
26	ARP flooding	S_{19}	10.30/11.18	81.67.45.12	Localhost	correlated
42	**Meta-Alert**	**{ S_4, S_1, S_{19} }**	**8.40/11.18**	**81.67.45.12**	**Localhost**	**{3, 17, 26}**

Figure 5 : Example of Alert Fusion

Besides this article, we can find relevant information in another article published in 1990 regarding Data Fusion and Alert Fusion. This article was written by Waltz and Llinas, determining the principal criteria for a system fusing data. However, it is unlikely that a system will meet all of the following requirements. According to this article, "the *more of these criteria a system meets, the more useful in alert fusion it is likely to be*":

- Distinguish parameters of interest
- Distinguish among different objects in space and time
- Sample the data and events of interest with sufficient frequency
- Provide accurate measurements
- Provide access to both raw and correlated data
- Use a common format (a single data format)

We have surveyed all the process of the Data Correlation, but the part about Alert Fusion is not in the chapter about Data Correlation, because this method has not already been implemented and needs an intelligent algorithm for its implementation. Therefore, we can highlight the fact, that in order to benefit the use of Data Correlation, we need to automate day tasks and to use special intelligent sensors, who are able to perform relevant tasks for reducing the time that analysts spend to make parameters, to analyse and to reply to the alerts. Thus, we will study this topic within the next sub-section.

B. AUTOMATED INTELLIGENT SECURE AGENT

First of all, most of the reasoning of this part is deriving from chapters about the "*Autonomous Agents*" of the following modules: **P00485** & **P00486**. Consequently, the main goal of this method is to **automate tasks as much as possible**. Thus, you surely want to know, what are agents, Why is this a powerful concept and useful for intrusion detection? What are its advantages?

So, according to the definition of Michael Wooldridge from the University of Liverpool: "***An Agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible autonomous action in that environment in order to meet its objectives***". Thus, in our case, Agents will perform security tasks in monitoring all the traffic to and from the host (detecting intrusion on hosts). We can use them, in order to gather data, fuse and correlate them. They can fulfil daily tasks in order to aid analysts in their work (can complete the job of correlating engine and sensors). In addition, we can use them to process audit by using mobile agents and also to share information about suspicious events and then determine when to be more vigilant.

Our ultimate goal, is to use an autonomous agent like Letizia (an autonomous agent for web Browsing: http://lieber.www.media.mit.edu/people/lieber/Lieberary/Letizia/Letizia.html) but in our case, for advising the administrator about a suspicious modifying of the network.

We generally use a multi-agent instead of an individual one, it consists of a community of agents (distributed network detection) acting autonomously, continuously, independently, concurrently with other agents in order to achieve their goals. Furthermore agents are operating in parallel with users; we can consider an Agent like an assistant or a helper who advises analysts about important decisions. It is also important to notice that analysts will delegate daily tasks to the Agents. The mains advantages of Agents are:

- Adapt and Anticipate to the needs of the analysts.
- Learn new method and new techniques from its experiences.
- Can display Human characteristics as inference, creativity ...
- Web and networks are domains well suited for autonomous agents.
- Compute time that otherwise be wasted.
- Fault tolerant behaviour
- Take initiative.
- Scalability and Flexibility
- Provide explanation of its actions.
- Can be added and removed from a system without altering other components.

- Can be implemented in the language that is best suited for completing its tasks.

So, now that we have underlined the advantages of using an autonomous intelligent agent to aid us in our intrusion detection work, we have to explain how we may build an architecture based on the Agent use, likewise how to administer them and how to be sure of their work?

Consequently for our needs, we will use a community of agents (for performing specific functions), and each agent will be independent, run entity present at some host, each host can have many agents and agents are only dealing with their own host. Moreover, we will need a storage base of all the information that has been gathered by the agents, this storage will facilitate the sharing of information between agents.

For more details about the architecture, we can survey the architecture of **AAFID** (Autonomous Agent for Intrusion Detection) developed by Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, David Isacoff, Eugene Spafford, Diego Zamboni. This agent uses: Filters (to get data), Transceivers (control and data processing over the agents), Monitors (oversee operation of transceivers, communicate with the UI (user interface)). Although the main advantage of using this layered architecture is the scalability, there is an important weakness in this process, if the communication between the entities is disrupted, the system will stop working. Furthermore, in this architecture Monitor entities are single points of failure.

Now, we will design an example of Intrusion Detection architecture based on the use of an autonomous agent, this design is inspired by the architecture of agents like AAFID, MAs.

Figure 6 represents a usual network architecture using agents and an intrusion system. In this design, the role of these agents is to sense the environment, collect the data for a further correlation, to perform a fusion of the information, in order to reduce the processing time of the correlation and above all to reduce the amount of alerts and to sense the system, if there is attempt of attacks against Mediator, Correlation base or Storage base. Moreover, it is not designed in *Figure 6*, but agents are able to directly communicate with the analyst in the event of critical attacks. Mediators, are similar to the Transceivers in the AAFID model; however, they also ensure that the agents were not imperiled by an attack. Mediators are dependent on the correlation base, which computes the Data Correlation processing and then provides the alerts to the analyst, also advises the analyst about a likely response to the attack. In spite of that, the analyst will be the only one to respond to a misdemeanour. Although the system can prevent an attack, detect an attack, and advise about a relevant response, only the analyst is able to reply to an attack.

Regarding the storage base, its role is to audit the router, it also contains all the intrusion scenarios anticipated by the analyst and the information collected by the agents, all this information is spread to the agents.

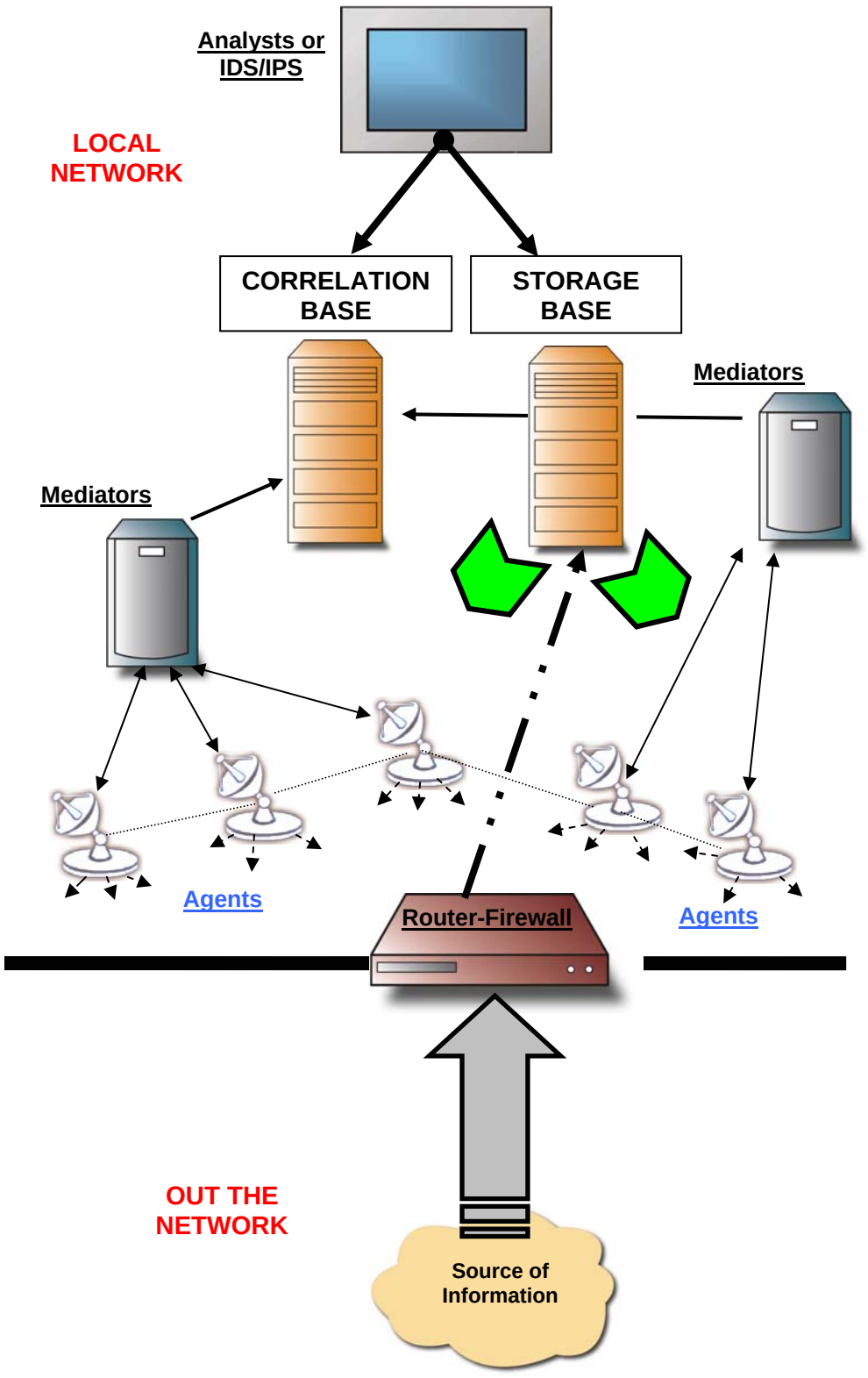

Figure 6 : Intrusion Detection architecture based on autonomous agent.

C. HYBRID METHOD: IMMUNOLOGICAL APPROACH FOR NETWORK DETECTION

Within this part, we will study a brand new approach for solving the chronic flaws of the network detection system. This approach is an analogy between the **human-immune system** and the computer security, why this analogy? Because the problem that the immune system solves, is the same as the one we want to solve for computing.

Thus, we already know that our immune system fights all day long to defend our body against harmful infections. So, we can probably reuse some of its techniques to apply them to the computer security issue. Furthermore, as we noticed in the chapter about IDS/IPS, we have an imperfect security and it can be apposite to use the human-immune system, because it is appropriate for dealing with the imperfect. Thus, all through this study we will explain how can work this analogy, and we will provide an example of an intrusion detection architecture based on immune methods.

First of all, it is important to highlight that the development of such a system would incorporate several elements of current security systems, thereby augmenting their resources. Apart from that the main advantages of the immune system are:

- Multi-layered: Many different layers are providing their own security mechanism (Skin, Physiological, Innate immune-system, Adaptive immune response).
- Disposability: None of components of the system are essential; any component can be replaced or supplied by others.
- Autonomously: The system doesn't need any management or relief.
- Adaptability: Adapt its prevention action, functions of the non-self intrusion.

The immune system protects us from any unsafe intrusion; this system is based on the identification of any foreign cell. In order to perform this, the immune system uses a similar technique to the pattern-matching recognition and distinguishes two important kinds of cells, the "**self**" and the "**non-self**". So, in immunology we called "self" cells of the body and "non-self" anything external of the body, and it is the main facet of the immune system. In addition, each immune system is considered as unique, we are not vulnerable to the same infections and not in the same way. Besides, the immune system can be viewed as a distributed intrusion detection system with the use of Lymphocytes, which are acting like agents (independent detectors) and are able to detect non-self cells thanks to the shape of their receptors (identification implemented as binding). Likewise, the protection of the self cells is more specific due to the learning and memory.

Moreover, the system can initiate a primary response in the case of the infection of a non-self already identified, also able to detect non-self cells that it has never encountered before by performing Anomaly Detection.

Although the analogy provides us with several advantages, there are also some limitations in the analogy. The immune system is too autonomous and sometimes makes errors (which can be critical for our system). Moreover all individuals will be vulnerable to the same pathogens, because each individual has a unique immune system. Likewise, an immune system is not "*formally specified*", so by definition it cannot be correct, consequently it is difficult to transfer its structure for computers, which are by definition the most formally specified tool. Thus, we have to re-use some methods used in the immune system and not make a total transfer of the overall system.

Once, we have understood how this system works, we have to respond to one question, if we desire to use the system as the immune system, we have to define what we can consider as self in our networks and similarly what we can define as non-self. I suggest that we can use in our network a certain method of identification each time that a packet is spread on the network. We can apply an identification process in order to tag this packet as a member of our system, and it is obvious that the identification tag will be different in the other systems. This tag can be a bit added to the TCP/IP packet. Another possibility is to use a database containing the normal behaviour of our network, and every time that an irregular event happens, we analyse it to see if it's an attempt to intrude or the normal development of the system.

For a relevant example of intrusion detection system based on the analogy with the human-immune system, we can quote the one developed by Stephanie Forrest from the University of New Mexico. She suggests considering each host of our system as a cell, with a network of mutually-trusting computers being the individual. Thus, we can use a set of agents for monitoring the state of each host into the system and when an anomaly occurs, the problematic "cell" could be isolated. If the source of the anomaly is outside the system, the agents must provide a primary aid to the non-contaminated hosts and then summon specific agents which will fight against the malicious machine for the sake of the network.

Figure 7. We apply this apposite architecture to the preview one that we have designed for the autonomous agents. This figure is representative of all the different techniques that we studied in the last chapter and above all in this chapter. The architecture which is represented in *Figure 7*, melt the use of Data Correlation, autonomous agent and immunological approach. This kind of architecture cannot be applied currently, notably due to a lack of relevant development in the field of the agent and especially in the field of the immune analogy. However, this architecture responds surely to need that we have in the domain of intrusion detection, because we can easily detect an accident, respond to it, reduce the problem of false alerts and then give more time to the analyst.

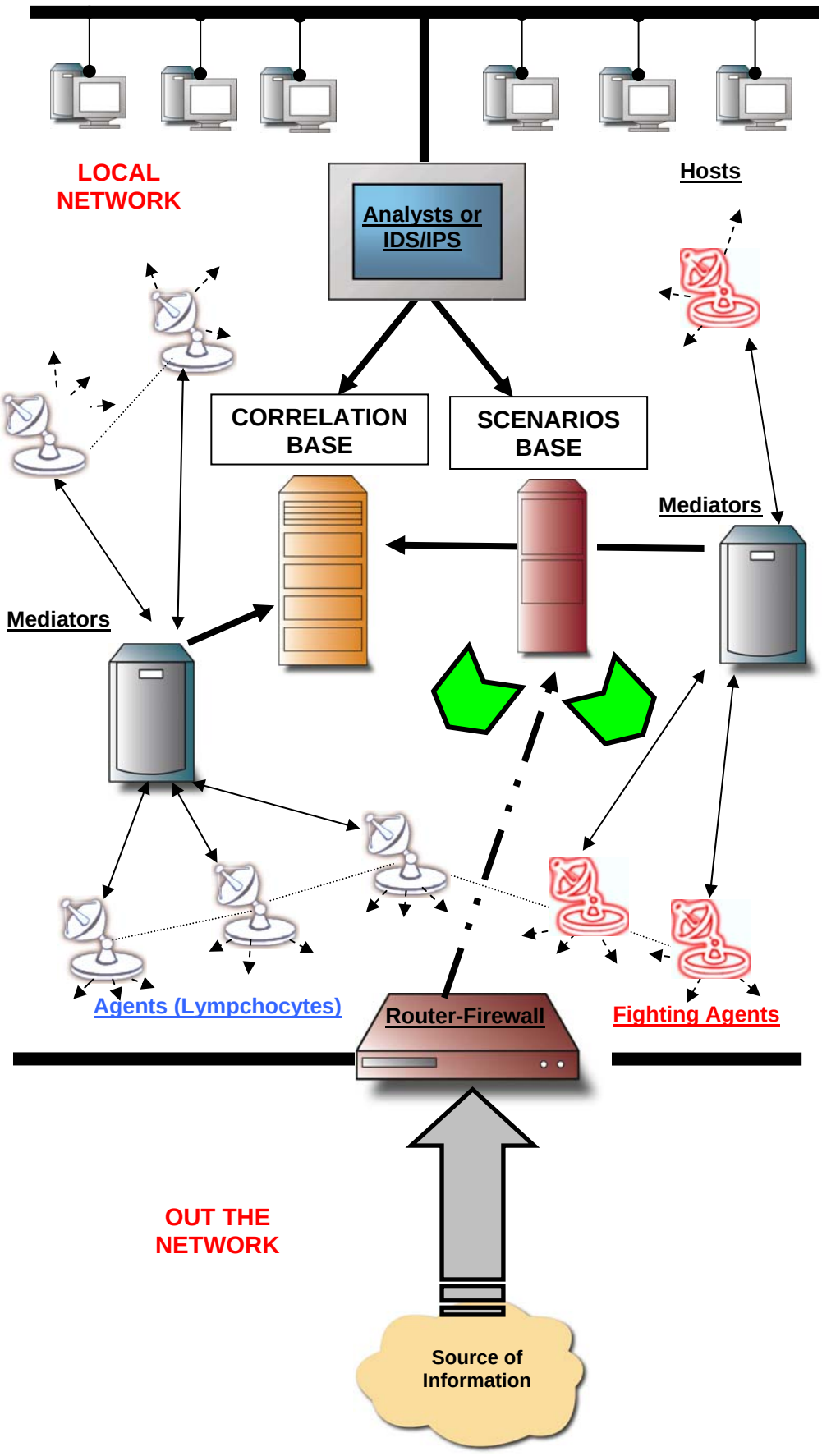

Figure 7 : Intrusion Detection architecture based on human-immune analogy and agents

VII. DATA-MINING

Within the last chapters, we have underlined that the administration of an Intrusion Detection system can quickly overwhelm the analyst by the number of alerts he needs to review. In order to solve that, we have studied new methods for improving the use of IDS/IPS. Although we have provided an interesting intrusion detection architecture using all the techniques that we have described, we need to explain what knowledge can be applied to our architecture, for the interplay between the different engines (agents, correlation base ...). We have to know how our agents would perform. How would we display the data? What are the best queries to highlight that data?

Above all we need knowledge to put reasoning in our correlation. **We don't want to protect the data, but the correlations among them**. All these requirements led us to investigate the application of Data Mining to this issue. Furthermore, the idea to apply Data Mining methods to computer security and intrusion detection came from a likely relationship between these two fields that some teachers from the module P00485 highlighted last semester.

Data Mining gives us "*facts*" that are not evident to human analysts; furthermore Data Mining enables inspection and analysis of huge amounts of information, we can also predict information about classified work from a correlation with unclassified work. Likewise, we can detect hidden information based on a conspicuous lack of events. Thus, we will focus our survey only on **Classification** and we will give a short explanation of the **Clustering**.

Figure 8. Represents the ideal use of the Data Mining methods in our Correlation base, according to Eric Bloedorn from the MITRE corporation.

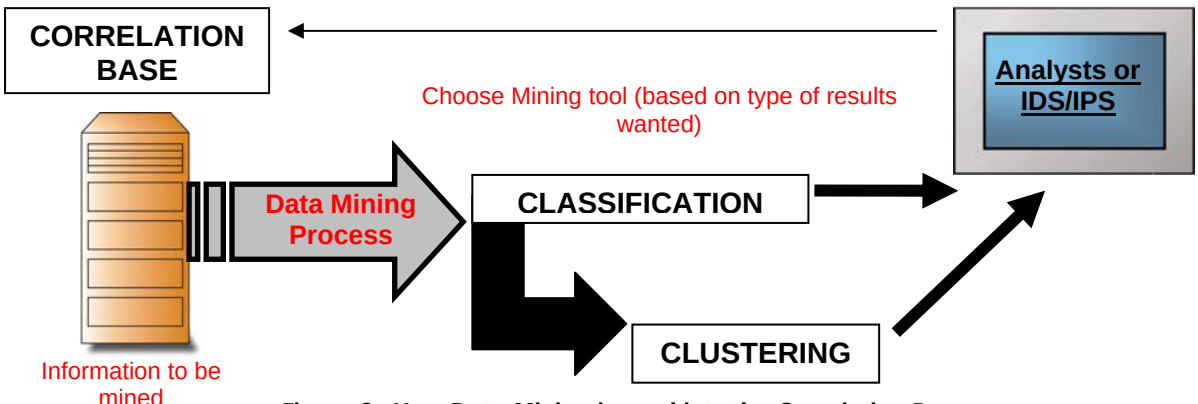

Figure 8 : How Data Mining is used into the Correlation Base.

A. CLASSIFICATION

The Classification method ***refers to the problem of attempting to predict the category of categorical data by building a model based on some predictor variables*** (source www.anwsers.com).

In the application of Classification to intrusion detection, we have to find a way to separate data into pre-defined groups. We will separate our set of data into two groups, **one representing the normal behaviour of our system** (defined with rules) and **the other representing the intrusive behaviour of our system,** also called abnormal behaviour asserting an intrusion in our system or an abnormal modification of our resources. Apart from that, Classification reduces false positives and negatives.

Thus the Classification method maps a data item into one of several predefined categories, the ideal application of this method in intrusion detection is to gather sufficient "**normal**" and "**abnormal**" audit data for a user or a program, then to apply a Classification algorithm to determine which data is belonging to a normal class or abnormal class. In order to be able to release this separation, we need a set of training data (data items where a group is known) for making it possible to recognize each data item from one of the groups. The goal of learning is to create a Classification model, called a classifier, which will predict with the values of its input attributes whether the data that we are analysing have a normal behaviour or an abnormal behaviour. (*Figure 9* sums up the mechanism of the Classification methodology).

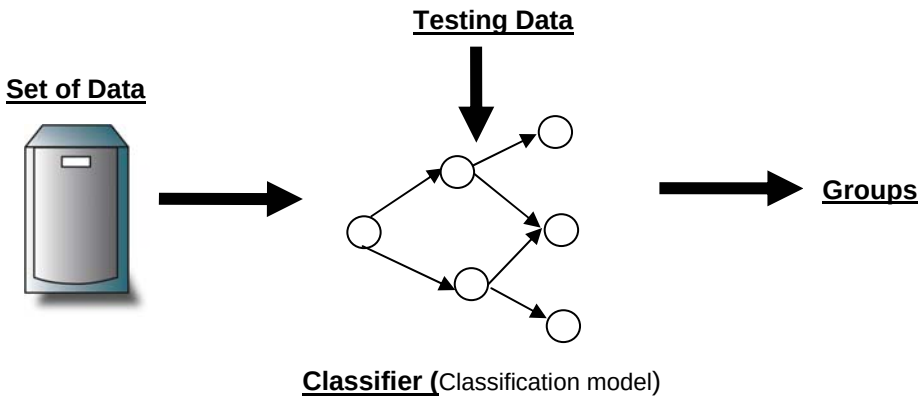

Figure 9 : Sketch of the Classification mechanism.

Figure 9. The set of data comes from an analyst who had manually reviewed cases, the testing data is one of the most important elements in order to achieve a good classifier performance, the quality of the testing data is the function of the number of relevant examples recorded and the attributes used to describe them.

In order to perform the Classification we might use suitable algorithms, as:

- Extensions to linear discrimination :
  - Neural Networks: "*Computer architecture modelled upon the human brain's interconnected system of neurons. Neural networks imitate the brain's ability to sort out patterns and learn from trial and error, discerning and extracting the relationships that underlie the data with which it is presented* ".(source Columbia University Press)

- Decision trees: Techniques that give in output the form of logical models. Consists of *nodes* where attributes are tested, the outgoing *branches* of a node correspond to all the possible outcomes of the test at the node, likewise one rule is generated for each *leaf*.(**J48** algorithm)

- Classification Rules: Is an alternative to decision trees, the precondition of a rule is a series of test just like the tests at nodes in decision trees. For example : **Nnge** (Nearest-neighbor-like algorithm using non-tested generalized examplars), builds a kind of "hypergeometric" model, including if-then rules, uses a similar approach to the "IB"s algorithms. Another algorithm like **Ridor** (Riple Down Rule learner) based on the generation of exceptions for the defined rules and the iterations of these exceptions for the best solution.

- Density Estimators :
  - **Naïve Bayes**: Simple and successful probabilistic classifier based on Bayes' Theorem, as we have learnt in Chapter VI. This theorem represents a theoretical background for a probabilistic approach to inductive-inferencing Classification problems. The main goal of the Naïve Bayes is to classify objects (data item ...). Thus, this method is based on probability assumptions which have no bearing in reality, but they can be trained most of the time very efficiently.
  - Hidden Markov Model: Model based on the Markov chain process, the goal is to determine hidden parameters.

Therefore, the Classification method is evidently convenience. The only disadvantage of this method is the training of the model; this testing can be long and needs apposite pre-recorded information. Besides, we don't know yet what is the most adaptable algorithm for intrusion detection, thus we will test all the algorithms in order to find the most suitable (probably Decision Rules/decision Trees). Finally, the main advantage is when we have separated the different groups "normal" and "abnormal" well, then with the testing (also called training sequence); the system will be able to predict unknown attacks or unknown events.

B. CLUSTERING

In Data Mining, we use Clustering, for arranging objects into groups, which is a natural and necessary skill that we all share. This method can be applied to a biological Classification of species or a Classification of events.

Thus, the Clustering method is an unsupervised machine learning technique for finding patterns in unlabeled data with many dimensions. Our ultimate aim is to find natural groupings of similar alarm records. Records that are far from any of these clusters indicate unusual activity that may be part of a new attack. Therefore, Clustering is well suited for classifying log data and detecting abnormal behaviour in the logs. As we already know, the normal amount of normal connection data is overwhelmingly larger than the number of intrusions. Thus the population of a normal cluster should be much larger than that of an intrusion cluster, and we may classify these clusters as “normal” or “abnormal” according to their population.

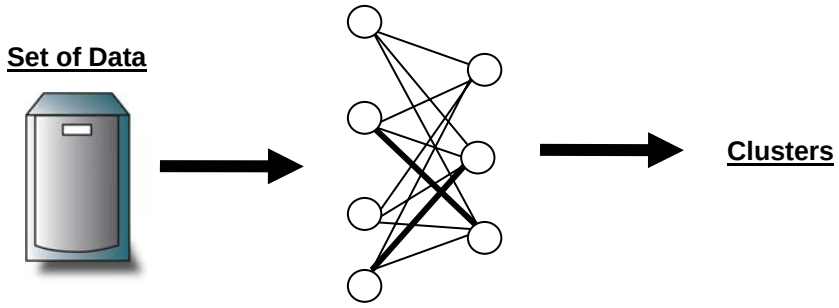

Figure 10 : Sketch of the Clustering mechanism

In addition, in order to classify our data into groups, we can use the following **attribute** which are specific of the network traffic:

- **Date**
- **Time**
- **Protocol**
- **Ip source**
- **Ip destination**
- **Port source**
- **Port destination**
- **Size of the packet**

The algorithm which is the most suitable for Clustering is "**K-means** Clustering" also called **k-nearest-neighbour**. K-means is able to automatically partition a set of data into a reasonable number of clusters, so as to classify the **instances** (attempts at modification of our system, each little point on *figure 11*) into the normal behaviour group and the abnormal behaviour group.

Apart from that, this algorithm is really popular (even though for the moment, no effective implementation of this algorithm was released) because its time and space complexity is relatively small. We can divide the algorithm mechanism into the basic following steps:

- Initialization: Select an initial partition with k **instances** from the set of data and make them initial cluster centres of the Clustering space.
- Assignment: Assign each instance to each closest centre.
- Updating: Replace each centre with the mean of its members.
- Iteration: Repeat "Assignment" & "Updating" till an optimum value is found or till that the system is stabilized.

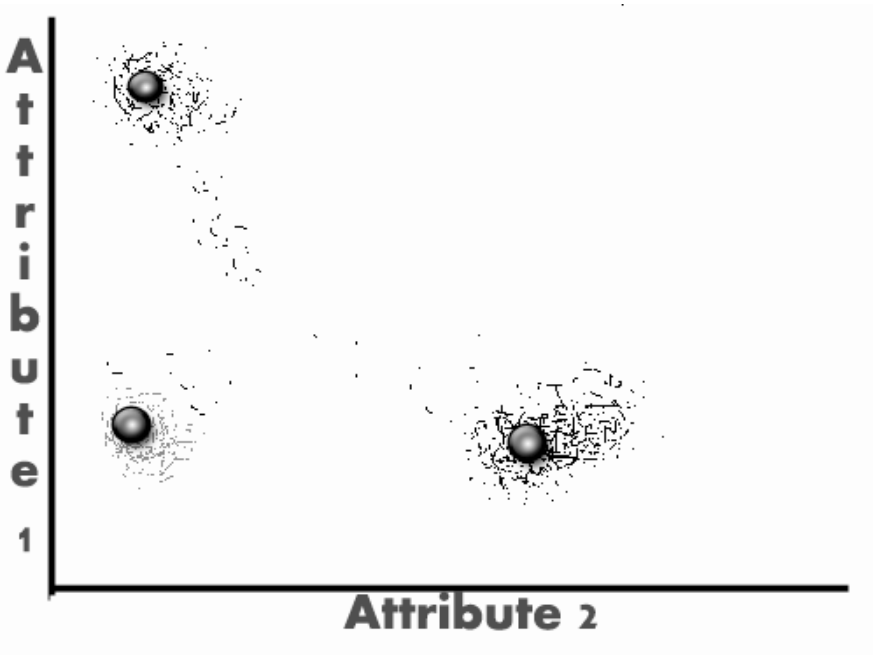

Figure 11 : Simple k-means with 3 clusters and 2 attributes

Unfortunately, it is very difficult to implement or to build a model for intrusion detection based on this algorithm, because the initialization of the k instances is made randomly, so we are not sure that the selection of our two groups will be apposite, and our log will be well separated into normal behaviour and abnormal behaviour. Therefore, the Classification method seems to be better suited for network intrusion.

According to a survey of Marcus J.Ranum for ICSA Labs in 2003, Classification methods or algorithms derived from them **will be able to detect around 90% of intrusion with a 1.1% of false alarm rate (positives and negatives), instead of 10% with basic techniques.** Therefore Classification and Clustering methods are promising techniques for intrusion detection.

VIII. IMPLEMENTATION

Within this chapter, we will describe the different steps of the analyser. How was it implemented? What resources were needed? What are the objectives?

First of all, the program implemented is an analyser of log. Our main goal with this analyser is to determine what algorithm is the most suitable for intrusion detection between the following Data Mining Classification algorithm: J48, NNge, Naive Bayes, Ridor (described in last chapter). The analyser, also performs Classification, because it gives in output a file called “LogOutput” with the log content classified, further the specification if an instance is normal or abnormal.

The program has been developed in **Java**, because the main advantage with Java is to modularize the entire system, then it becomes easier to add or remove features. Even though this analyser is just a demonstration of the creation of a classifier and then its log analysis, we can speculate reusing it for the implementation of the biggest tool like an IDS based on Data Mining. About the GUI, it was implemented with “swing” and we know that Java is not well suited for interface development, but our main aim was to develop a program relevant and useful to survey our needs.

Besides, Java also allows us to import Data Mining classes from a software called **Weka** (Ian H. Witten and Eibe Frank 1999-2005. *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*). Weka is a collection of machine learning algorithms in Java for solving real-world Data Mininig issues. Moreover, Weka integrates a large panel of algorithms (Classification, Clustering, Regression ...) and tools. We can consider Weka as our algorithm resources.

Furthermore, in order to test, train our Classifier, we needed specific log files. We finally found out available trained log files, on the website of Lincoln Laboratory of the MIT and their group of research in intrusion detection (DARPA).

	Start Date	Start Time	Duration	Serv	Src Port	Dest Port	Src IP	Dest IP	Attack Score	Name
1	01/27/1998	00:00:01	00:00:23	ftp	1755	21	192.168.1.30	192.168.0.20	0.31	-
2	01/27/1998	05:04:43	67:59:01	telnet	1042	23	192.168.1.30	192.168.0.20	0.42	-
3	01/27/1998	06:04:36	00:00:59	smtp	43590	25	192.168.1.30	192.168.0.40	12.0	-
4	01/27/1998	08:45:01	00:00:01	finger	1050	79	192.168.0.40	192.168.1.30	2.56	guess
5	01/27/1998	09:23:45	00:01:34	http	1031	80	192.168.1.30	192.168.0.40	-1.3	-
7	01/27/1998	15:11:32	00:00:12	sunrpc	2025	111	192.168.1.30	192.168.0.20	3.10	rpc
8	01/27/1998	21:53:17	00:00:45	exec	2032	512	192.168.1.30	192.168.0.40	2.95	exec
9	01/27/1998	21:58:21	00:00:04	http	1031	80	192.168.1.30	192.168.0.20	0.45	-
10	01/27/1998	22:57:53	26:59:00	login	2031	513	192.168.0.40	192.168.1.20	7.00	-
11	01/27/1998	23:57:28	130:23:08	shell	1022	514	192.168.1.30	192.168.0.20	0.52	guess

Figure 12 : Log example

About the log training, the main difficulties were to choose relevant information in the log files. We based our selection on the attribute selection that we did in last chapter. So our selection is :

- **Duration (Length of a connection)**
- **Service used**
- **Port source**
- **Port destination**
- **Ip source**
- **Attack score**

http://www.ll.mit.edu/IST/ideval/data/1999/training/week2/index.html

About the architecture of the program; as already explained in the Classification sub-section, we need to provide a Dataset and to build a classifier. So, the first part of the implementation was to build a classifier and after the calculation of a model, apply this model to new log files to make statistics.(*Figure 13*, example of the classifier model for Ridor algorithm).

```
RIpple DOwn Rule Learner(Ridor) rules
--------------------------------------

Class = intrusion attempt  (75927.0/6808.0)
          Except (port_dest = '(22-2274.5]') and (length = short) => Class = normal  (4031.0/0.0) [2000.0/0.0]
          Except (serv = ftp-data) and (ip_src = external) => Class = normal  (178.0/0.0) [82.0/0.0]
          Except (ip_src = internal) and (length = short) and (port_dest = '(14720-19027]') => Class = normal  (69.0/6.0) [27.0/3.0]
          Except (ip_src = internal) and (length = short) and (serv = ecr/i) => Class = normal  (53.0/0.0) [14.0/0.0]
          Except (ip_src = internal) and (length = short) and (port_dest = '(2274.5-7542]') => Class = normal  (78.0/12.0) [36.0/3.0]
          Except (ip_src = internal) and (length = short) and (serv = eco/i) => Class = normal  (40.0/0.0) [28.0/0.0]
          Except (ip_src = internal) and (length = short) and (port_dest = '(12632.5-13510.5]') => Class = normal  (19.0/0.0) [5.0/0.0]
          Except (ip_src = internal) and (length = short) and (serv = ftp) => Class = normal  (12.0/0.0) [6.0/0.0]
          Except (ip_src = internal) and (length = short) and (port_dest = '(31815-inf)') => Class = normal  (14.0/3.0) [3.0/0.0]
          Except (port_src = '(2844.5-24034]') and (port_dest = '(22-2274.5]') => Class = normal  (26.0/0.0) [6.0/0.0]
          Except (ip_src = internal) and (length = short) and (port_src = '(2844.5-24034]') => Class = normal  (4.0/0.0) [3.0/0.0]

Total number of rules (incl. the default rule): 12
```

Figure 13 : Ridor Classifier model

We can see that, the classifier provide us a model that it computed during the building of the classifier. In this model, we diversify two groups, the normal behaviour of the system and the abnormal behaviour. Some models provide us the rules that it has figured out for detectin abnormal behavior in this log. For example : Nnge model

```
class intrusion attempt IF : length in {short, long} ^ serv in {ftp} ^
port_src in {'(2844.5-24034]','(24034-inf)'} ^ port_dest in {'(20.5-22]'} ^
ip_src in {external}  (63)
```

This rule means, that an instance will be detected as an intrusion only, if the duration of the connection is long, if the service used is “Ftp”, if the port source is betwwen 2844 and 24034, and if the port destination is between 20 and 22 and the Ip source of the connection is external to our network. Thus, a model will provide this kind of rules, but they are specific to our log files and to the algorithm, that’s why some algorithms can detect a connection as an intrusion and others cannot.

In the statistics, we highlighted above the rate of intrusion detection in the log and the rate of false positive; thanks to this result we will be able to study the different algorithms and then deduce which one is most suitable for intrusion detection. *Figure 14* sums up the architecture of the analyser. Otherwise, further information, regarding the analyser are providing with Appendice C.

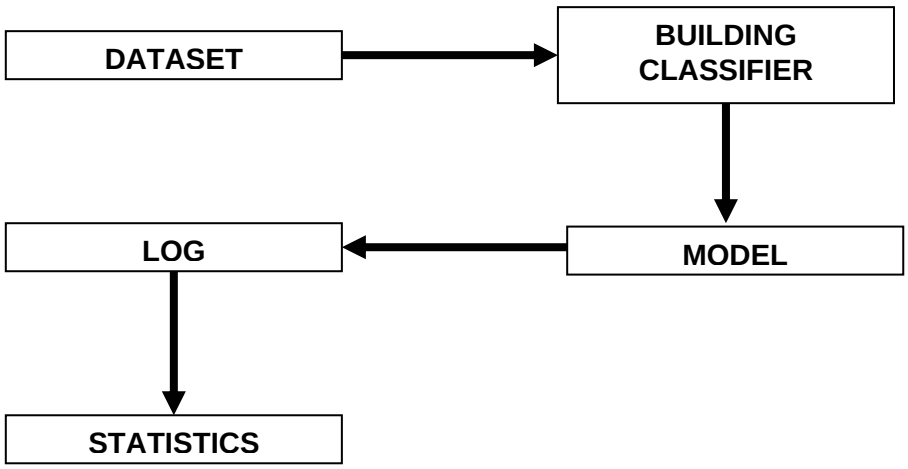

Figure 14 : Analyser architecture

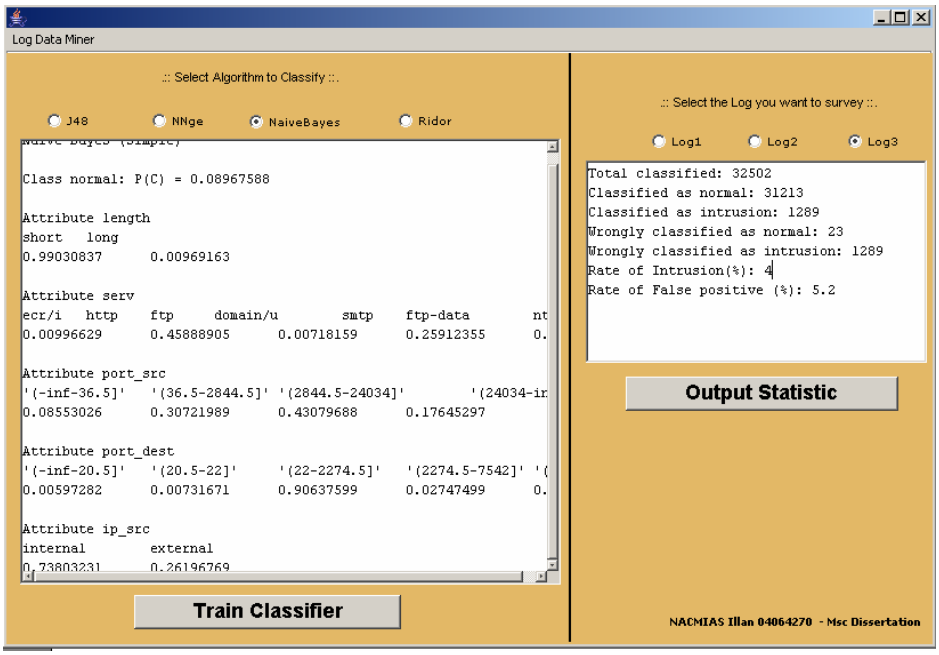

Figure 15 : The program

IX. EXPERIMENTATION

In this chapter, we will survey four Data Mining algorithms relating to the Classification method. We will use the analyser of the last chapter and we will discuss the statistical results during the processing of the classifier on the log files.

A. MODEL COMPARAISON

Within this part, we will compare the different models that we got during the classifier building process. It is interesting to compare those models because we can underline what are the main rules in order to distinguish a normal behaviour and an intrusion behaviour. Moreover, we will only focus on the comparaison between the different definition of an intrusion.

- **J48 :** An instance will be define as an attempt to intrude, whether :

```
ip_src = internal
|   port_dest = '(2274.5-7542]'
|   |   length = long: intrusion attempt
|   port_dest = '(7542-7603]': intrusion attempt
|   port_dest = '(8036-11383.5]': intrusion attempt
|   port_dest = '(11651.5-12632.5]': intrusion attempt
|   port_dest = '(13510.5-14720]': intrusion attempt
|   port_dest = '(14720-19027]'
|   |   length = long: intrusion attempt
|   port_dest = '(19027-20558]': intrusion attempt
|   port_dest = '(20753.5-21614.5]': intrusion attempt
|   port_dest = '(21825.5-21918.5]': intrusion attempt
|   port_dest = '(22274.5-24283.5]': intrusion attempt
|   port_dest = '(24283.5-24626.5]': intrusion attempt
|   port_dest = '(24626.5-24694.5]': intrusion attempt
|   port_dest = '(24694.5-30911.5]'
|   |   serv = http: intrusion attempt
|   |   serv = ftp: intrusion attempt
|   |   serv = domain/u: intrusion attempt
|   |   serv = smtp: intrusion attempt
|   |   serv = ftp-data: intrusion attempt
|   |   serv = ntp/u: intrusion attempt
|   |   serv = time: intrusion attempt
|   |   serv = ident: intrusion attempt
|   |   serv = finger: intrusion attempt
|   |   serv = telnet: intrusion attempt
|   |   serv = pop3: intrusion attempt
|   |   serv = irc: intrusion attempt
|   port_dest = '(30911.5-31014]': intrusion attempt
|   port_dest = '(31014-31815]': intrusion attempt
ip_src = external
|   serv = ecr/i: intrusion attempt
|   serv = ftp
|   |   length = long: intrusion attempt
|   serv = domain/u: intrusion attempt
```

```
|   serv = ntp/u: intrusion attempt
|   serv = eco/i: intrusion attempt
|   serv = time: intrusion attempt
|   serv = ident: intrusion attempt
|   serv = pop3: intrusion attempt
|   serv = urp/i: intrusion attempt
```

This model bases its Classification on the distinction on the Ip address internal or external of the network. It is interesting to notice that for the internal Ip address, it focuses on the port source and the service used, whereas for the external Ip address it is only focusing on the service used, it means that for this model the attributes port sources, destintaion and length of connection are meaningless, whether we know that an Ip address is external and we know the service used.

- **NNge:**

```
* class intrusion attempt IF : length in {short} ^ serv in {ftp-data} ^ port_src in {'(-inf-365]'} ^ port_dest in {'(14720-19027]','(24626.5-24694.5]','(24694.5-30911.5]','(30911.5-31014]','(31014-31815]'} ^ ip_src in {internal}
* class intrusion attempt IF : length in {short, long} ^ serv in {ecr/i,ftp} ^ ip_src in {external}
* class intrusion attempt IF : length in {long} ^ serv in {ftp-data} ^ port_src in {'(-inf-36.5]'} ^ port_dest in {'(22-2274.5]','(2274.5-7542]','(7542-7603]','(14720-19027]','(24626.5-24694.5]','(24694.5-30911.5]','(31014-31815]'} ^ ip_src in {internal}
* class intrusion attempt IF : length in {short,long} ^ serv in {ftp-data} ^ port_src in {'(-inf-36.5]'} ^ port_dest in {'(8036-11383.5]','(11651.5-12632.5]','(13510.5-14720]','(19027-20558]','(20753.5-21614.5]','(21825.5-21918.5]','(22274.5-24283.5]','(24283.5-24626.5]','(31815-inf)'} ^ ip_src in {internal}
```

Within this model, we can highlight that when an Ip address is external and the service used is Ftp or ecr/i, our intsnace will be classified as an intrusion.

- **Naive Bayes :**

```
Attribute length
short 0.99632528

Attribute serv
ecr/i http  0.99247838

Attribute port_src
'(-inf-365]' 0.84901961

Attribute ip_src
External 0.99373562
```

This model just provides statitsics about the training of the building, we just put the most apposite results. So, regarding these results, we can underline that 99% of the intrusion attempts are from short connection to our system, also 99% of the intrusion are from the

following service : ecr/i http. Moreover 84% of the intrusion attempts have for port source a port between 0 and 365. Finally 99% of the intrusion attempts are from external ip address.

- **Ridor :**

```
Class = intrusion attempt
 Except (port_dest = '(22-2274.5]') and (length = short)
 Except (serv = ftp-data) ^ (ip_src = external)
 Except (ip_src = internal) ^ (length = short) ^ (port_dest = '(14720-19027]')
 Except (ip_src = internal) ^ (length = short) ^ (serv = ecr/i)
 Except (ip_src = internal) ^ (length = short) ^ (port_dest = '(2274.5-7542]')
 Except (ip_src = internal) ^ (length = short) ^ (serv = eco/i)
 Except (ip_src = internal) ^ (length = short) ^ (port_dest = '(12632.5-13510.5]')
 Except (ip_src = internal) ^ (length = short) ^ (serv = ftp)
 Except (ip_src = internal) ^ (length = short) ^ (port_dest = '(31815-inf)')
 Except (port_src = '(2844.5-24034]') ^ (port_dest = '(22-2274.5]')
 Except (ip_src = internal) ^ (length = short) ^ (port_src = '(2844.5-24034]')
```

Anew in the model we can notice, that when an Ip address is external it is directly classified as an intrusion(especially if the service used is Ftp). Because this model, only provides the rules which are classified as an intrusion. Thus the rules hereinbefore underline that almost all the instances are classified as an intrusion, likewise the rules are quiet similars to the rules of NNge.

B. MEASUREMENTS

This sub-section sums up, the different measurements computed by the analyser (see appendice B), the main results are representing in the figure underneath and all this results will be discuss in the next sub-section.

Furthermore, “mean of rate of intrusion” means the percentage of attempt to intrude in the log files. The “mean of false” represents the percentage of false alerts in the log files. There is also the “classifier’s mean length of building” and also the “mean Classification time” which represents the mean tim needed to classify the log files, it is an apposite feature in order to perform intrusion detection in real-time.

	J48	NNge	Naive Bayes	Ridor
Mean of rate of Intrusion %	1.95	4.52	5.084	2.50
Mean of false %	1.73	2.615	5.147	2.523
Classifier’s mean length of building	1371ms	3455 ms	821 ms	11917 ms
Mean Classification time	1235ms	1920ms	1332ms	1239ms

Figure 16 : Mean rate of intrusion and false alerts per algorithm relating the analyser

C. DISCUSSION

First of all, we chose these algorithms because they are representative of the main techniques in Classification, J48 represents the “tree technique”, NNge and Ridor the “rules technique” and Naive Bayes the “probabilistic technique”. Apart from that we also tried other algorithms like IBk(IB1 ...) but the processing time was too long surely due to the power of our computer and we often got the message as we have not enough memory to perform this operation.

Otherwise these algorithms are really quick to compute for the building of our classifier and to compute the statistical results. Furthermore, in *Figure 16*, we sum up the measure that we did regarding the classifier’s building time and the mean Classification time. From those results we can notice that J48 and Naive Bayes algorithms are the most quick to compute, this feature is very interesting, for the analyst who want to know the algorithm more suitable for real-time detection.(Similarly, the results of *Figure 16* are depending of the processing time of our computer, so they may change at every execution of the application).

Thus, we based the statistics computation on the following equations:

$$\textbf{False positive rate} = \frac{\textbf{False positive*100}}{\textbf{Classified as an Intrusion + False positive}}$$

$$\textbf{Rate of Intrusion} = \frac{\textbf{Classified as an Intrusion*100}}{\textbf{Classified as normal + Classified as an Intrusion}}$$

Regarding the statistics results we can underline that the percentage of intrusion in our log is very low (8.42 % see Appendice B) for the maximum. In *Figure 16*, we notice that our results are similars to our conclusion in Chapter VII, execpt for the Rate of false positive, but our bad results (50% of false positive by log) are due to the training of our classifier, we did not train the classifier enough, above all because of the lack of processing resources.

About the Dataset and the normal number of connections into our logs, we have selected them in order to have around 4% of intrusion by log file. So, regarding *Figure 16*, NNge seems to be the most suitable for the detection, because the rate of intrusion detected by its algorithm is noticeably close to 4%.

In spite of that, another criteria is the mean of false alerts, as well the mean rate of false positive as the mean rate of false negative.

$$\textbf{\textit{Mean of false alerts}} = \frac{\textbf{\textit{Mean of False positive + Mean of false negative}}}{\textbf{\textit{Total Classified}}}$$

More this rate is low, and more the detection is efficient. Regards our results, the low rate is for J48 (1.73%), even though NNge and Ridor are really close with only (2.5%) of false alerts.

Therefore, due to our result we can deduce what algorithm or what technique is well suited for detecting intrusion. The method well suited is the Classification Rule/Decision tree method. This both methods are quite similar and regarding to our survey are perfect for detecting intrusion. However, there is still, some limitations in the computation process of the results, we will discuss of these limitations within the next sub-section.

D. LIMITATION

Unfortunately, there are several limitations in the building of this program. First of all, due to the power of the computer which did the calculation, we cannot use a bigger Dataset. Our results are meaningful, but they are not so accurate. In order to have more exact rules and model we should need to train our classifier with different log files. The major limitation in that problem is not the size of the dataset, DARPA provides 9Go of log files, but our computer has not the capacity to process all this information (that's why the size of our dataset is only 6.5mo). Still about the power of computer, we did not survey an algorithm like IBk, because it will spend around 75 mins computing our statistics.

Similarly, our training is not so accurate, because the result of the false positive is still high around 50% per log, and with more data (we have data, but not enough time to process all the data that we have, one solution is to distribute the log computation).

Another limitation is the log files. It could be better to develop a program which transforms every kind of log files into a common format to make it useful for our program. It was the idea at the beginning, but it was too difficult to develop, there are too many different kinds of log files, and it is difficult to find at least one of them in order to build the architecture of a common log file. By chance we found the DARPA association with their logs, but the log files of current IDS, Ftp, or ssh do not have the same format. So, the limitation is that we cannot use the classifier model on other logs in order to consult the rate of intrusion and false alerts.

Moreover, another limitation in our experimentations is that the model released is not usable for all the networks. It means that we cannot reuse its rules to configure an IDS. It is important to notice that the model produced by our analyser is specific to the content of our Dataset, thus specific to the DARPA network.

Finally, one of the major limitations was in the building of the Dataset, because we had to choose what data in the log files was interesting for us, and then transform some data from numeric state into nominal state, which is more useful for the Data-mining algorithm. So, we did this operation, and we transformed the duration of a connection into short and long instead of 00:0015, 01:04:57 ... Even though this process allows us a greater speed in the processing of our classifier, it also seems a bit biased, because by this process we may have obscured other Classifications in the model.

E. FUTURE WORK

Despite these limitations, we achieved our goal, and we deduced that the Classification method well suited for intrusion detection is the Classification Rules/Decision trees. Now we are able to understand why this method is actually being tested in some IDS.

We can continue to analyse the algorithms and write a Classification guidebook, in order to advise analysts in their implementation of intrusion detection systems.

However, it is more interesting to try to develop a real intrusion detection engine, using Data Mining algorithms. This survey gave us enough knowledge in the field of Data Mining applied to intrusion detection in order to build a real engine (even though we will need lots of time and lots of resources).

Besides, we can also develop a data correlation engine, likewise based on Data Mining and based on our analyser. Our analyser just looks into the log and then provides statistics, but we can modify it to get the abnormal instances in output. Then, to develop a way to respond in time to an intrusion, our analyser just monitors log files for the moment, but with more resources (time and investment) we can develop a real engine which performs intrusion detection.

X. CONCLUSION

Following the reading of this dissertation, we may still have some questions regarding the network detection. The one that you obviously want to ask is, when will all these new techniques be used by the IDSs/IPSs? We don't know yet, we hope as soon as possible, but for the moment, all of these techniqes have to be tested in order to be commercialised. Data Mining methods have lots of advantages, but also some limitations, we need to find the best tradeoffs between our expectations and capital budget. Furthermore, in this dissertation we have provided apposite results relating the different algorithms of Data Mining most suited for Intrusion Detection and we have also provided an sketch of a likely perfect intelligent intrusion detection architecture. Then, we hope that all these results will interest new students, or that some of them carry on the study of the network security issue in Oxford Brookes University, and even constitute a group of researchers of this topic in Brookes, who are able to develop in depth and implement some of techniques and key ideas introduced in this document.

XI. APPENDICE A

Pseudocode for explaining the inner working of the Classification algorithms :

- **Decision Tree(J48) :**

```
buildtree returns node_ptr
  IF examples all of the same class
  THEN  return a pointer to a LEAF where the class is the one with the most
              examples and the certainty factor is the number
              in the class divided by the total number of examples.
  ELSE
      choose criteria C
      IF choosing process fails
      THEN return pointer to LEAF as above
      ELSE
        split_examples( C, examples, yes_examples {VAR} , no_examples {VAR}
)
        LET yes_b = buildtree( yes_examples )
        AND no_b  = buildtree( no_examples )
        return a pointer to a BRANCH
                       with criteria   =  C
                            yes_branch =  yes_b
                            no_branch  =  no_b
      END {if choosing fails}
  END {if examples of same class}


classify( example: example_type, root: node_ptr ) returns class_type
  current_pos = root
  WHILE  current_pos is not a leaf
     IF  example satisfies current_pos^.criteria
     THEN   current_pos := current_pos^.yes_branch
     ELSE   current_pos := current_pos^.no_branch
  { ends with current_pos on a leaf }
  return current_pos^.class
```

- **Classification rules (Ridor and NNge) :**

```
For each class C
        Init Q to the instance set
        While Q  contains instances in class C
            Create a rule R with an empty left-hand side that predicts
            class C
            until there is no more attributes to use, do
        For each attribute A not mentioned in R, and each value v,
                Consider A=v to the LHS of R
                Select A and v to maximize the accuracy
        Add A=v to R
Remove the instances cobered by R from Q.
```

- **Naives Bayes :**

```
Init C with one component.
k   k0
repeat
        Add k new mixture components to C, initialized using k
        random examples from T.
        Remove the k initialization examples from T.
        repeat
           E-step: Fractionally assign examples in T to mixture components,
           using C.
           C-step: Compute maximum likelihood parameters for C,
           using the filled-in data.
           If log P(H|M) is best so far, save C in Cbest.
           Every 5 cycles, prune low-weight components of C.
        until log P(H|M) fails to improve by ratio E/C.
        C   Cbest
        Prune low weight components of C.
        k   2k
until log P(H|C) fails to improve by ratio _Add.
Execute E-step and C-step twice more on Cbest, using examples
from both H and T.
Return Cbest.
```

XII. APPENDICE B

Statistical results of the analyser computation :

- **J48 :**
  - **Log1 :**

    Classifier's building time: 1371 ms
    Time for Classification: 1933.0 ms
    Total classified: 58495
    Classified as normal: 57987
    Classified as intrusion: 508
    Wrongly classified as normal: 0
    Wrongly classified as intrusion: 508
    Rate of Intrusion(%): 0.9
    Rate of False positive (%): 50
  - **Log2 :**

    Classifier's building time: 1371 ms
    Time for Classification: 921.0 ms
    Total classified: 14248
    Classified as normal: 13718
    Classified as intrusion: 530
    Wrongly classified as normal: 11
    Wrongly classified as intrusion: 516
    Rate of Intrusion(%):3.71
    Rate of False positive (%): 49.33
  - **Log3 :**

    Classifier's building time: 1371 ms
    Time for Classification: 851.0 ms
    Total classified: 32502
    Classified as normal: 32091
    Classified as intrusion: 411
    Wrongly classified as normal: 9
    Wrongly classified as intrusion: 397
    Rate of Intrusion(%): 1.26
    Rate of False positive (%): 49.13

- **NNge :**
  - **Log1 :**

    Classifier's building time: 3455 ms
    Time for Classification: 2974.0 ms
    Total classified: 58495
    Classified as normal: 57570
    Classified as intrusion: 925
    Wrongly classified as normal: 0
    Wrongly classified as intrusion: 925
    Rate of Intrusion(%): 1.58
    Rate of False positive (%): 50
  - **Log2 :**

    Classifier's building time: 3455 ms
    Time for Classification: 1272.0 ms
    Total classified: 14248
    Classified as normal: 13708
    Classified as intrusion: 540
    Wrongly classified as normal: 25
    Wrongly classified as intrusion: 540
    Rate of Intrusion(%): 3.79
    Rate of False positive (%): 50
  - **Log3 :**

    Classifier's building time: 3455 ms
    Time for Classification: 1512.0 ms
    Total classified: 32502
    Classified as normal: 31777
    Classified as intrusion: 725
    Wrongly classified as normal: 23
    Wrongly classified as intrusion: 725
    Rate of Intrusion(%): 2.23
    Rate of False positive (%): 50

- **Naive Bayes :**
  - **Log1 :**

    Classifier's building time: 821 ms
    Time for Classification: 2043.0 ms
    Total classified: 58495
    Classified as normal: 56839
    Classified as intrusion: 1656
    Wrongly classified as normal: 0
    Wrongly classified as intrusion: 1656
    Rate of Intrusion(%): 2.83
    Rate of False positive (%): 50
  - **Log2 :**

    Classifier's building time: 821 ms
    Time for Classification: 1112.0 ms
    Total classified: 14248
    Classified as normal: 13048
    Classified as intrusion: 1200
    Wrongly classified as normal: 25
    Wrongly classified as intrusion: 1200
    Rate of Intrusion(%): 8.422
    Rate of False positive (%): 50
  - **Log3 :**

    Classifier's building time: 821 ms
    Time for Classification: 841.0 ms
    Total classified: 32502
    Classified as normal: 31213
    Classified as intrusion: 1289
    Wrongly classified as normal: 23
    Wrongly classified as intrusion: 1289
    Rate of Intrusion(%): 4
    Rate of False positive (%): 50

- **Ridor :**
  - **Log1 :**

    Classifier's building time: 11917 ms
    Time for Classification: 1923.0 ms
    Total classified: 58495
    Classified as normal: 57862
    Classified as intrusion: 633
    Wrongly classified as normal: 0
    Wrongly classified as intrusion: 633
    Rate of Intrusion(%): 1.08
    Rate of False positive (%): 50
  - **Log2 :**

    Classifier's building time: 11917 ms
    Time for Classification: 941.0 ms
    Total classified: 14248
    Classified as normal: 13590
    Classified as intrusion: 658
    Wrongly classified as normal: 13
    Wrongly classified as intrusion: 646
    Rate of Intrusion(%): 4.615
    Rate of False positive (%): 49.53
  - **Log3 :**

    Classifier's building time: 11917 ms
    Time for Classification: 851.0 ms
    Total classified: 32502
    Classified as normal: 31915
    Classified as intrusion: 587
    Wrongly classified as normal: 11
    Wrongly classified as intrusion: 575
    Rate of Intrusion(%): 1.806
    Rate of False positive (%): 49.48

XIII. APPENDICE C

This appendice contains a description of the cd content and all files related to the implementation and the statistical survey. The main directory is :

- Workspace : This directory is the exact copy of the java workspace that I have implemented. It contains, all the files needed to launch and use the program, as well as the log files for the training and the test.

About the Weka© algorithms, they are distributed under the GNU General Public License (GPL), license is also enclosed in the cd.

Moreover a JAR file directory is stored at the cd root. This JAR file should be launched using a Java Runtime environment's version later than 1.5. The command for launching jar files is:

java –jar jar_name.jar.

Furthermore, we have to be in D:\JAR File in order to be able to launch the program. Otherwise, it is required to copy, the JAR File directory on your computer, then to place on the correct directory where you have copied the directory and then launch the program with the correct command.

Apart from that, about the inner working of the analyser, it is important to notice that after launching the analyser, we have to click on "Train Classifier" in order to build and train the classifier. Only after this step, we will be able to get the statistical results by clicking on "Output statistic". Then, we will also allow to choose the log file that we want to survey.

Furthermore, this program provides several outputs :

- Classifier.txt is the file used by weka for the Classification process.
- BuildClassifier.rtf is the classifier model.
- LogOutput.txt is the result of the Classification process. This file contains all the content of the analysed log, further the specification if an instance corresponds to a normal behaviour of the system or an anormal behabiour.

XIV. REFERENCES

- Eibe Frank, Ian H.Witten, (1999-2005). *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*.
- DARPA, Lincoln Laboratory, Massachusetts Institute of Technology http://www.ll.mit.edu/IST/ideval/data/1998/1998_data_index.html
- Lawrence A.Gordon, Martin P.Loeb, William Lucyshyn and Robert Richardson, CSI/FBI, (2004). *Computer Crime and security survey*.
- Symantec Internet Security (March 2005).Threat Report volume VII, trends for July 04-December 04.
- Lawrence Allen, Rajesh Cherukuri, Stephanie Forrest, Alan S.Perelson, (1994). *Self-Nonself discrimination in a Computer* presented by University of New Mexico.
- Patrick D'haeseleer, (1996). *An immunological approach to change detection : theoretical results*, presented by University of New Mexico.
- Stephanie Forrest, Anil Somayaji, (2000). *Automated response using system-call delays*, presented by University of New Mexico.
- Stephanie Forrest, Steven A.Hofmeyr, (1999). *Immunity by design: An artificial immune system*, presented by University of New Mexico.
- Stephanie Forrest, Steven A.Hofmeyr and Anil Somayaji, (1998). *Principles of a computer immune system*, presented by University of New Mexico.
- Dennis L.Chao, Stephanie Forrest, (2003). *Information Immune systems*, presented by University of New Mexico.
- Eric Boedorn, Alan D.Christiansen, William Hill, Clement Skorupka, Lisa M.Talbot, Jonathan Tivel, (2001). *Data Mining for intrusion detection : How to get started*, presented by The MITRE corporation.
- Wenke Lee, Salvatore J.Stolfo, (1998). *Data Mining approaches for intrusion detection*, presented by Columbia University.
- Keith Skinner, Alfonso Valdes, (2001). *Probabilistic Alert Correlation*, presented by SRI International.
- Zakia Marrakchi, Ludovic Mé, Ricardo S.Puttini, (2003). *A bayesian Classification model for real-time intrusion detection*, presented by Supélec Rennes France.
- Anita K.Jones, Robert S.Sielken, (1999). *Computer system Intrusion Detection : A survey*, presented by University of Virginia.

- Susan M.Bridges, German Florez, Rayford B.Vaughn, (2002). *An improved algorithm for fuzzy Data Mining for Intrusion Detection*.
- Yuko Maruayma, Jun-ichi Takeuchi, Kenji Yamanishi, (2005). *Data Mining for security*.
- Paul Dokas, Levent Ertoz, (2003). *Data Mining for Network Intrusion Detetcion*, presented by University of Minnesota.
- Grant Anderson, Remco Bouckaert, Ben Cleland, Kurt Driessens, Dale Fletcher, Eibe Frank, Mark Hall, Geoff Holmes, Ashraf Kibriya, Richard Kirkby, Althon Lin, Brent Martin, Bernhard Pfahringer, Peter Reutemann, Gabi Schmidberger, Tony Smith, Ian H. Witten. *WEKA Data Mining Software in Java.* University of Waikato. http://www.cs.waikato.ac.nz/ml/weka/.
- Winn Schwartau(1999), *Time Based security*.
- E. Waltz and J. Llinas. Artech House, Norwood, MA (1990), *Multisensor Data Fusion*.
- Guy G.Helmer, Vasant Honavar, Les Miller, Johny S.K.Wong, (1998). *Intelligent Agents for Intrusion Detection*, presented by Iowa state University.
- *NCSA, The National Centre For Supercomputing Applications,* http://www.ncsa.uiuc.edu/
- CERIAS, *Autonomous Agents for Intrusion Detection*, http://www.cerias.purdue.edu/about/history/coast/projects/aafid.php
- The Honeynet Project : http://project.honeynet.org/
- Philip Torr, *Rational Reasoning*, http://cms.brookes.ac.uk/staff/PhilipTorr/
- Module P00485 & P00486 Oxford Brookes University, http://cms.brookes.ac.uk/modules

XV. BIBLIOGRAPHY

- Kerry J.Cox, Christopher Gerg. Managing security with SNORT and IDS tools, (August 2004), isbn : 0-596-00661-6.
- Lance Spitzner, Honeypots Tracking Hackers, (September 2002), isbn : 0-321-10895-7
- Richard Bejtlich, The Tao of network security monitoring beyond Intrusion Detection, (July 2004), isbn : 0-321-24677-2
- Carl Endorf, Jim Mellander, Eugene Schultz, Intrusion Detection & Prevention, isbn : 0-072-22954-3.
- Bruce Schneier (2000). *Secrets and Lies: Digital Security in a Networked World.* John Wiley & Sons.
- Stephen Northcutt, Judy Novak, *Network intrusion detection*, isbn : 2-7117-4831-6