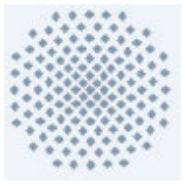
DEVELOPMENT OF AN ADAPTIVE LEARNING NETWORK-ATTACK DETECTION SYSTEM

Laura Herranz Embid

Supervisors: Antonio Cuevas
Uwe Fischer



Universität Stuttgart

ACKNOWLEDGEMENTS

After seven months in Stuttgart, I would like to thanks all the people that have made this incredible experience even more special, in particular to Iñigo Ucero Aristu, Patrick Kranzinger, Vanessa Isla Hernández and Alberto Bethencourt Alonso. I would like to say thank you too to my tutor, Antonio Cuevas, for his help.

I dedicate this study to my parents and to Agustín Fraile Poblador. I miss you all.

ABSTRACT

The proliferation of Internet and the increase of the number of network computers cause a raise of network attacks that attempt to confidentiality, integrity and availability of the computer infrastructures. Therefore Intrusion Detection Systems (IDSs) have become an essential part of today's security infrastructures. There exists different kind of IDS. The separation that interest us the most for this study is misuse and anomaly-based IDSs. The first of them detects and classifies attacks with predefined rules and the second checks how much traffic differs from "normal" traffic and adapts itself to know in each moment what is normal and what not. The goal of this study is to propose a new IDS for the Stuttgart's University network since the current one called Peakflow is a misuse IDS and can't detect novel attacks. Here it is proposed SPADE as new IDS. SPADE detects anomalies based in probabilities and decides through a threshold that adapts according with the last results. SPADE solves the problem of novel attacks but we will see that this isn't always very efficient because it can considerer abnormal traffic to be normal when the attacks are continuous or when there isn't enough traffic normal in order to calculate the probabilities correctly and introduce a high false alarm rate.

Key words: IDS, confidentiality, integrity, availability, attack, misuse IDS, anomaly-based IDS, adapt, novel attack, threshold, false alarm rate.

ZUSAMMENFASSUNG

Die Verbreitung des Internets und die steigende Zahl vernetzter Computer führt zu einer Zunahme von Angriffen, welche den Schutz vertraulicher Daten und die Integrität und die Sicherheit von Netzwerken gefährden. Deshalb wurden, Intrusion Detection Systems‘ (IDSs) zu einem essentiellen Bestandteil heutiger Sicherheitsinfrastrukturen. Es gibt verschiedene Arten von IDS. Der Schwerpunkt dieser Arbeit liegt im Bereich der ‚missbrauchsbeschränkenden‘ IDS, welche Angriffe an Hand vordefinierter Regeln erkennen und klassifizieren - und der auf Anomalitäten basierenden IDS, welche kontrollieren, wie weit der Datenverkehr von ‚Normalwerten‘ abweicht und sich selbst stetig an die veränderten Randbedingungen anpassen, um erkennen zu können, was ‚normal‘ ist und was nicht. Ziel ist es der Universität Stuttgart ein neues IDS vorzuschlagen, da das momentan benutzte System, ‚Peakflow‘, ein missbrauchsbeschränkendes IDS ist, das neuartige Angriffe nicht entdecken kann. Hierbei bietet sich SPADE als neues IDS an. SPADE erkennt Anomalitäten, indem es die Eintrittswahrscheinlichkeit mit einem Schwellenwert vergleicht, der sich nach den letzten Ergebnissen richtet. SPADE löst das Problem der Erkennung neuartiger Angriffe, wobei man allerdings beachten muss, dass es nicht immer sehr effizient arbeitet, da es anormale Netzwerkaktivitäten als normal einstufen kann, falls die Angriffe fortwährend andauern, oder aber die Anzahl der Fehlalarme sehr hoch liegt, falls die reguläre Netzwerknutzung nicht ausreicht, um die Eintrittswahrscheinlichkeiten korrekt zu berechnen.

Schlüsselwörter: IDS, Datenschutz, Integrität, Verfügbarkeit, Angriff, missbrauchsbeschränkende IDS, anomalitäts-basierende IDS, anpassen, neuartige Angriffe, Schwellenwert, Fehlalarmrate.

INDEX

I. INTRODUCTION 9
II. Attacks to the network 12
III. Intrusion Detection Systems (IDS) 15
a. IDS used in Stuttgart's University: Peakflow 15
1. How it works 15
2. Arquitecture 15
b. Neuronal networks: SOM 16
1. Data selection-normalization 18
2. Initialization 19
3. Learning 19
4. Decision 20
c. Probabilistic method: SPADE 20
1. Probability tables 21
2. Threshold. Adaptation 21
3. Output 21
d. Algorithm finally used 22
IV. Installation and get off the ground 23
V. Implementation 25
a. Place the IDS 25
b. Simulation of the attacks 28
1. IDSwakeup 28
2. NMAP 29
3. Spike 29
VI. Experiments 31
a. First experiment: attack detection from Internet 31
b. Second experiment: IDSwakeup attack between hosts 32
c. Third experiment: Nmap attack between hosts 34
d. Fourth experiment: Spike attack between hosts 36
e. Fifth experiment 38
f. Conclusions of the experiments 39
VII. Problems found 40
VIII. Conclusions and further work 40
IX. Resumen en español 41
a. Introducción y puesta en marcha 41
b. Experimentos 47
1. Detección de ataques desde Internet. 47
2. Ataques con IDSwakeup 47
3. Ataques con Nmap. 48
4. Ataque Spike 48
5. Uso de una base de datos. 49
c. Conclusiones 51
X. References 52

INDEX OF FIGURES

Figure 1: Number of new threats detected by Symantec from July 2002 to December 2007. 9
Figure 2: Cost of security violations. 10
Figure 3: Attack sophistication increasing in the last years. 12
Figure 4: Portscannig attack. 13
Figure 5: Example of DoS attack. 14
Figure 6: Peakflow solution network architecture 16
Figure 7: Networks components. 7.a: Collector: gather flow either directly from the router or by packet capture identify anomalous traffic and transfer findings to the controller. 7.b: Controller: aggregates and archives statistics and creates a network-wide view. 16
Figure 8: How a training neural network works. 17
Figure 9: Multilayered neural network. 17
Figure 10: Procedure for building profiles with SOM. 18
Figure 11: SOM learning scheme. 19
Figure 12: Gaussian neighbourhood function 20
Figure 13: Typical locations for an intrusion detection system. 25
Figure 14: IDS connected to a spanning port. 26
Figure 15: IDS connected to a hub 26
Figure 16: Scenario. 27
Figure 17: ROC curve of SPADE. 39
Figure 18: Alert management scenario 40

INDEX OF TABLES

Table 1: Malicious activity by country. 10

I. INTRODUCTION

The explosive growth of computer networks and particularly of the Internet has created many stability and security problems. In 2007, Symantec [1] detected 711,912 new threats compared to 125,243 in 2006 – an increase of 468 percent; this brings the total number of malicious code threats detected by Symantec to 1,122,311 as of the end of 2007.

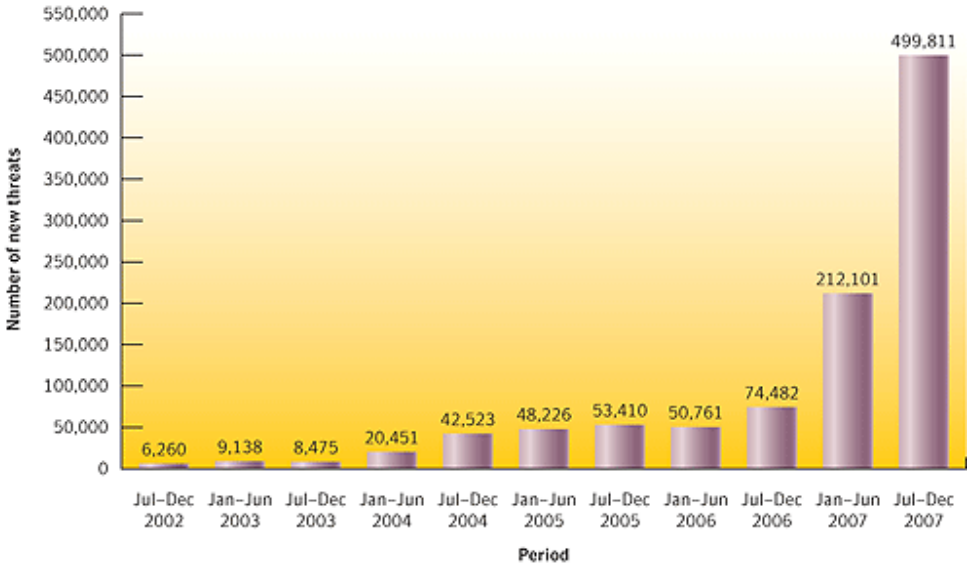


Figure 1: Number of new threats detected by Symantec from July 2002 to December 2007.

In order to have an idea in where the malicious traffic is more important, Symantec provides also a table with the top ten countries more affected by malicious traffic.

Current Rank	Previous Rank	Country	Current Percentage	Previous Percentage	Bot Rank	Command-and-Control Server Rank	Phishing Web Sites Host Rank	Malicious Code Rank	Spam Zombies Rank	Attack Origin Rank
1	1	United States	31%	30%	1	1	1	1	1	1
2	2	China	7%	10%	3	5	2	2	4	2
3	3	Germany	7%	7%	2	2	3	7	2	3
4	4	United Kingdom	4%	4%	9	6	7	3	12	5
5	7	Spain	4%	3%	4	19	15	9	9	4
6	5	France	4%	4%	8	13	6	11	7	6
7	6	Canada	3%	4%	13	3	5	4	35	7
8	8	Italy	3%	3%	5	10	11	10	6	8
9	12	Brazil	3%	2%	6	7	13	21	3	9
10	9	South Korea	2%	3%	15	4	9	14	13	10

Table 1: Malicious activity by country.

United States ranked first with a large margin. This is normal since malicious traffic usually affects computers that are connected to a high-speed broadband Internet and United States has the most established broadband infrastructure in the world. That is also the reason for the China's second and Germany's third place.

Attacks affect significantly to the performance of the network. A study from Information Sciences Institute of California shows that, for example, DDoS attacks causes DNS latencies to increase by 230%, and the web latencies to increase by 30% [2].

Besides, in a financial point of view, the failures caused by these attacks are translated into a huge cost. As it can be shown in the next figure, Virus induce a lost of more than 5 million dollars in 2004 [3].

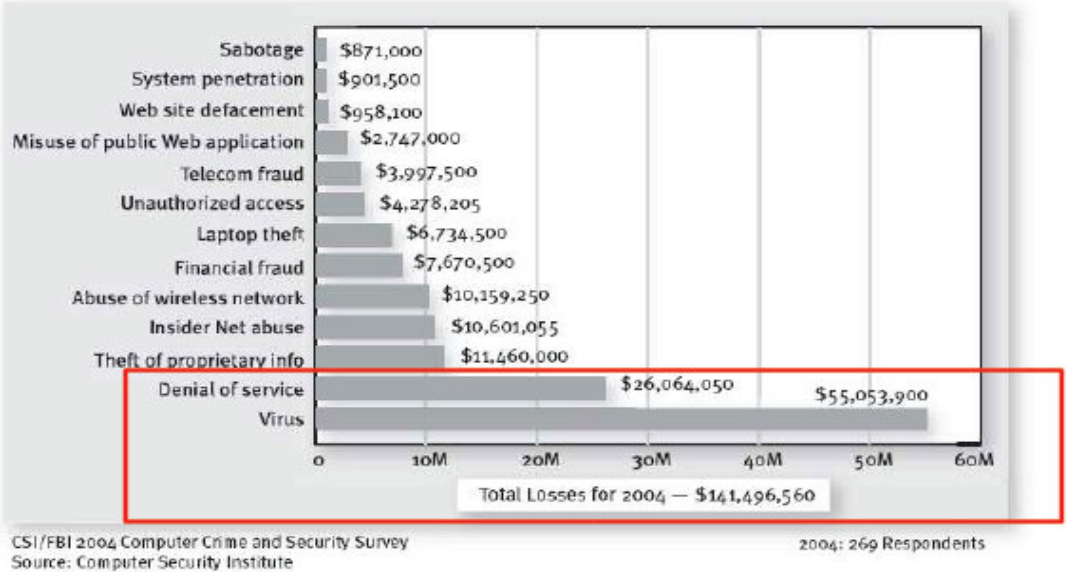


Figure 2: Cost of security violations.

The increasing and negative features of attacks make their detection a critical process. Therefore the Intrusion Detection Systems (IDS) are implemented. As it will be discussed later, IDS are software and/or hardware designed to detect unwanted attempts at accessing, manipulating, and/or disabling of computer systems, mainly through a network, such as the Internet. An intrusion detection system is used to detect several types of malicious behaviours that can compromise the security and trust of a computer system.

The simplest technique of intrusion detection is the misuse detection that's based on saved patterns of known attacks, that is to say, it compares the network connections features to the attack patterns provided by human experts. This technique isn't useful for unknown intrusions that appeared continuously in the network because the algorithms must be uploading manually for each new attack. The other technique that adapts much better to the evolution of the network's attacks is the anomaly detection one. This kind of technique defines what is normal and attempt to track deviations from the normal behaviour that are considered to be intrusions. Anomaly detection systems can be divided into adaptive and non-adaptive ones. Adaptation is related with the ability of the IDS to modify some of his parameters according with the past observations in order to follow the changes over the time of the normal-behaviour patterns.

In this study, a method of improving for the monitoring system of the University of Stuttgart (Peakflow) is implemented where the attack detection will be anomaly-based and adaptive. The work will begin with a brief introduction of the different attacks to the network, then the Peakflow system will be presented, the next step will be the demonstration of new techniques of attacks detection, next a short description of the scenario implemented, the experiments done and then, the conclusions.

II. Attacks to the network

There are multiple kinds of attacks and they are changing continuously as the attackers have more tools and knowledge to make them more sophisticated. The next figure shows this vertiginous rise [3]:

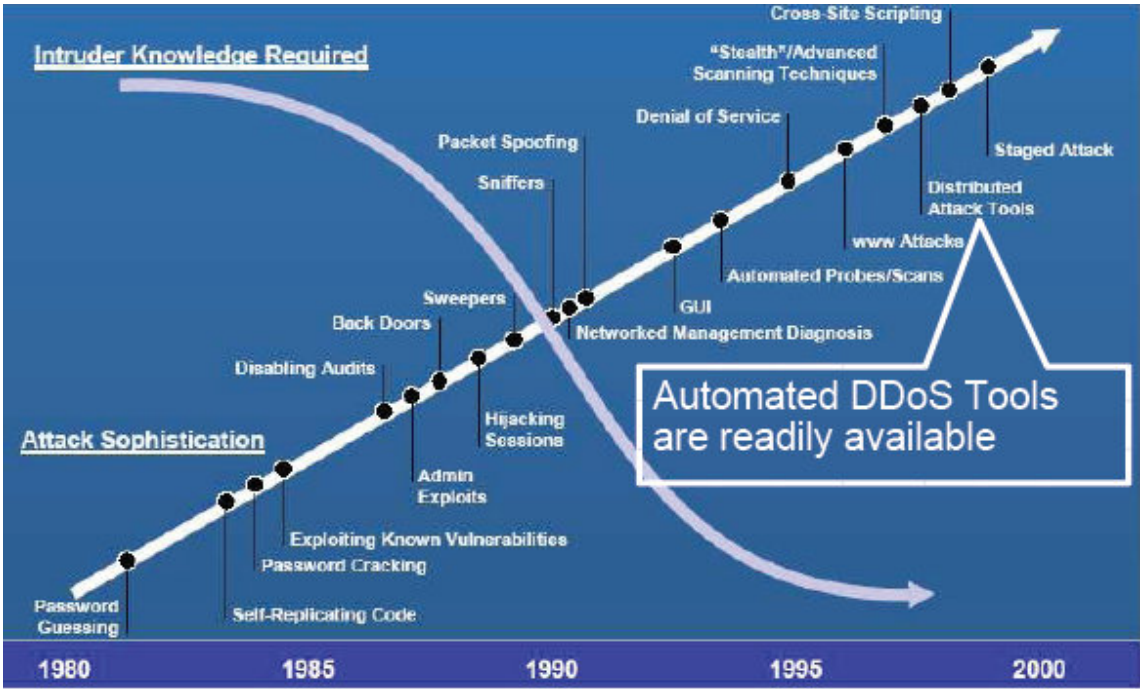


Figure 3: Attack sophistication increasing in the last years.

Attacks attempt to:

- Data integrity: trustworthiness of information resources.
- Data availability: information resources should be avalaible when they are needed.
- Data confidentiality: limited information access and disclosure to authorized users and preventing access by or disclosure to unauthorized ones.

Malicious traffic can be classified into multiple groups. Here, only three of them that have been considered the most important and useful for the following experiments:

- Worms: self-replicating computer program that uses a network to send copies of itself to other nodes. This traffic is usually detected by a regular expression called signature, for example the following signature will detect the worm Witty traffic [5]:

```
alert udp any 4000 -> any any (msg:"ISS PAM/Witty Worm Shellcode"; content:"|65
74 51 68 73 6f636b5453|";depth:246;classtype:misc-attack; sid:1000078;)
```

This expression is a kind of filter that alerts when the packet obeys some rules. It has been shown that the Witty Worm's payload is contained in a single UDP packet with a fixed source port of 4000 and variables destination port, source and destination IPs. The message that will appear in case of attack will be the one in inverted comas. The payload after the firsts 246 bytes should match with the content of the signature. A misc-attack is a classification of snort's attacks and means that it's a low priority one. Finally the sid code identifies uniquely snort rules.

- Portscanning: scans the network trying to find vulnerable hosts to be then attacked so it is useful for system administrators and other network defenders to detect portscans as possible preliminaries to a more serious attack.

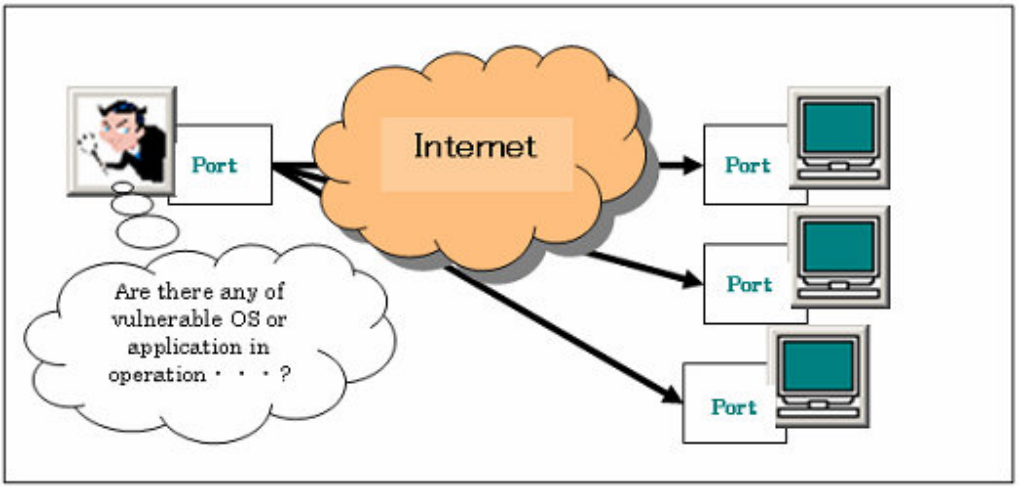


Figure 4: Portscannig attack.

- A denial-of-service attacks (DoS attack) or distributed denial-of-service attack (DDoS attack): attempt to make a computer resource unavailable with attacks from one computer (DoS) or from multiple computers (DDoS). One common method of attack involves saturating the target (victim) machine with external communications requests, such that it cannot respond to legitimate traffic, or responds so slowly as to be rendered effectively unavailable (view figure 5). Smurf, Ping of Death or Teardrop are other methods of a DoS attack.

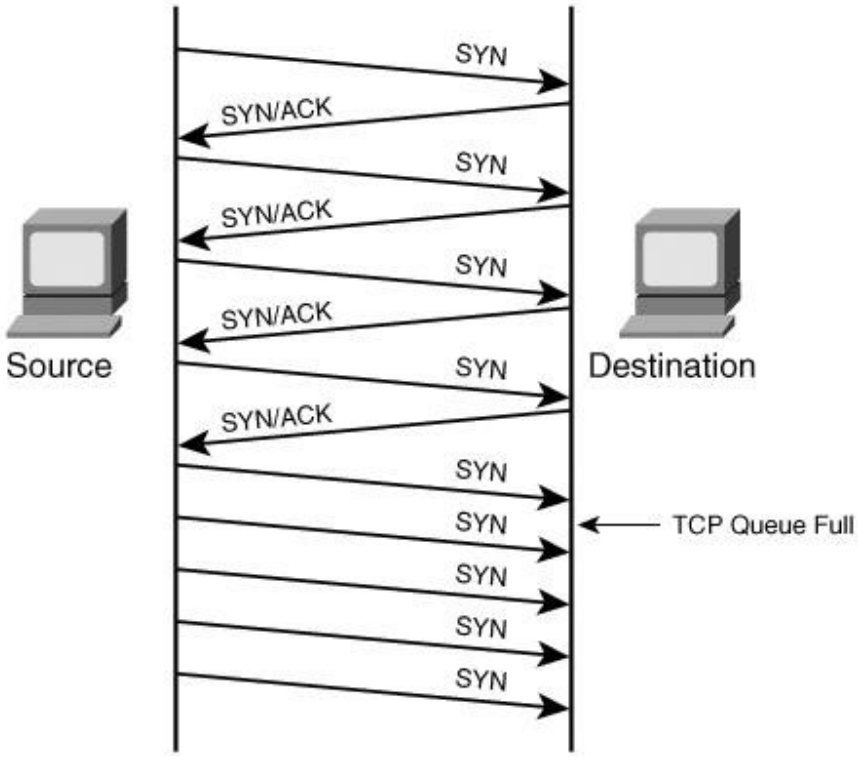


Figure 5: Example of DoS attack.

Malicious traffic infests the networks and it's a need to control them in order to protect network availability. The intrusion detection systems (IDS) are implemented for that purpose.

III. Intrusion Detection Systems (IDS)

An intrusion detection system (IDS) monitors network traffic in order to detect attacks to the network [9]. IDS can be located in strategic points within the network; these are called NIDS (Network Intrusion Detection Systems) or in individual hosts, called HIDS (Host Intrusion Detection Systems). As it has been said in the introduction, there are signature-based IDS where the attacks are well-known and defined and there's anomaly-based IDS that compares network traffic against an established baseline which identifies what is "normal" for that network: bandwidth, ports, IPs, … generally used.

a. IDS used in Stuttgart's University: Peakflow

The software used in the Rechezentrum of the University of Stuttgart (RUS) for monitoring traffic and detect attacks to the network is the Peakflow [4]. Peakflow is a signature-based IDS that creates static rules and compares traffic against these baselines to detect anomalous network behaviour.

1. How it works

Peakflow regularly samples network traffic statistics and routing data across an entire network in order to create a networkwide perspective. This can be done at a very high speed (up to OC-192 (10Gbps)). With that information, Peakflow builds graphs that display traffic and routing information, both real-time and historical.

Peakflow uses static rules in order to detect attacks to the network. This way to classify would be useful in a constant environment but, as it as been already explained, computer networks have a dynamic nature in a sense that information and data within them are continuously changing. The intrusions whose signatures are not in the database will be undetected.

2. Arquitecture

The Peakflow solutions are each made up of two network components: collectors and controllers. Collectors are used to obtain Netflow data from the monitored routers or sniff packets on a port span adjacent to these routers. They will also poll the routers using SNMP to obtain port names and link speeds and in they will establish BGP connections with the monitored routers to get reachability information. Controllers collect data from one or more collectors, aggregate and correlate the information and either generate alerts or store the data into databases. The collectors provide the GUI interface for user access. The diagram below shows the protocols that are run between routers and collectors, collectors and controllers, and users and controllers:

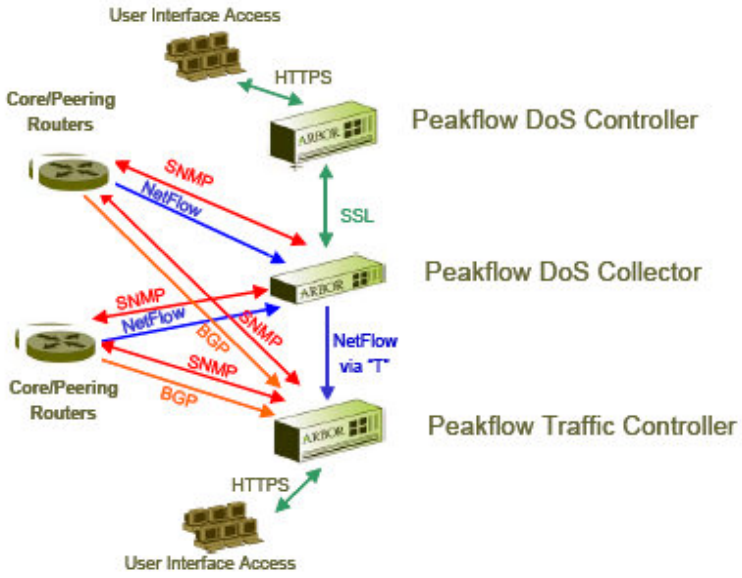


Figure 6: Peakflow solution network architecture

Figure 7: Networks components. 7.a: Collector: gather flow either directly from the router or by packet capture identify anomalous traffic and transfer findings to the controller. 7.b: Controller: aggregates and archives statistics and creates a network-wide view.

This system should operate in real time and adapt himself for detecting an intrusion accurately and promptly. For that the system needs to be able to generalize from previously observed behaviour to recognize similar future behaviour and future behaviours not identical to the past observed ones. In the next sections, two solutions will be presented

b. Neuronal networks: SOM

A neuronal network is able to capture complex (non linear) input/output relationships. They are based on the human brain: acquires knowledge through learning and his knowledge is stored within inter-neuron connections (weights). A disadvantage of neural networks is that they need to be trained with a large amount of data that allows generalization and it isn't always possible. An example is shown in the next figure where there are two inputs (features), in the neural network; the weights are calculated then adjusted calculating the minimum error with the desired output [10].

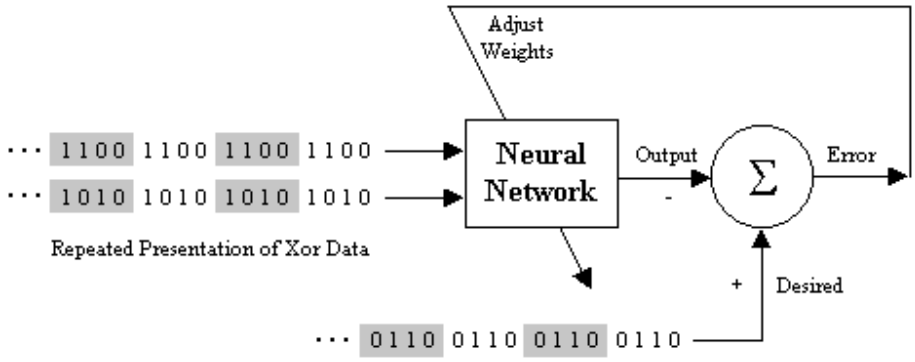


Figure 8: How a training neural network works.

The neuronal network can be composed of one or more layers as it can be seen in the figure below.

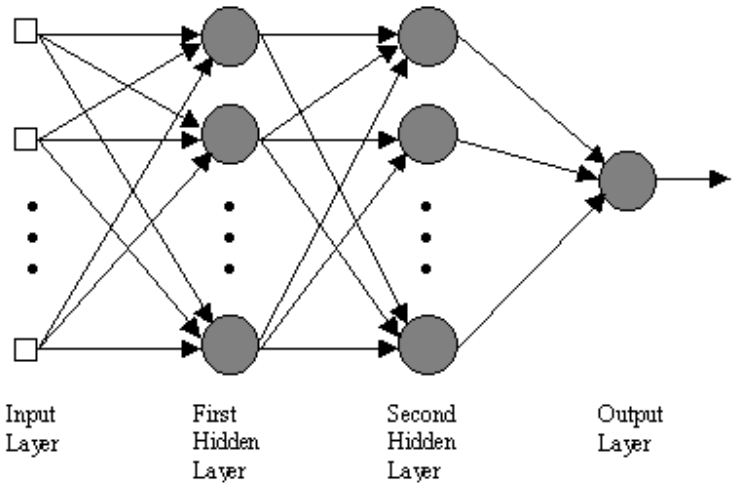


Figure 9: Multilayered neural network.

There are multiple types of neural networks but one that is used in attacks' detection is the Self-Organizing Map algorithm (SOM). SOM models data points from a complex input space into a lattice of neurons in the self-organizing map. In the next figure, the diagram of the model is shown [11]. Each box will be later discussed.

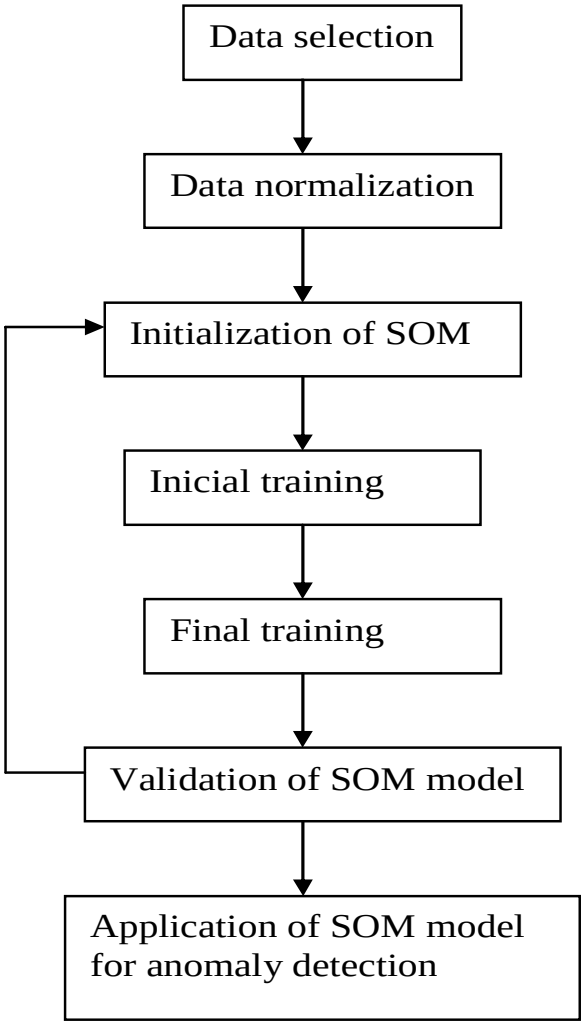


Figure 10: Procedure for building profiles with SOM.

1. Data selection-normalization

First, traffic data is captured and all the fields of the IP heather extracted. With the values of these fields, dimensions can be calculated, for example the duration of the connective or of the inactivity. These dimensions are collected for different classes of network traffic and then used to train the SOM. It's clear that the dimension with high variance will tend to dominate the map organization so; first of all, the dimensions should be normalized so that the variance of each of the dimensions in the training data is unity [6].

It's important to capture a sample data set that encompasses all possible range of possible types of normal traffic.

2. Initialization

The solution obtained by Self-Organizing Map (SOM) strongly depends on the initial neuron/cluster centres. However, all existing SOM initialization methods do not guarantee to obtain a better minimal solution. Generally, these methods can be divided in two classes: random initialization and data analysis based initialization classes.

3. Learning

In the learning phase, neurons are trained to model the input data points in the k-dimensional space. Each input is fed with all the neurons in parallel and the neuron that responds the best is selected and her values adjusted in order to respond even better with a similar input in the future. The neurons in the winner's neighbourhood are also adjusted for the same purpose. As it can be seen in the next figure only the winner and the neighbourhood (black and grey neurons) are touched.

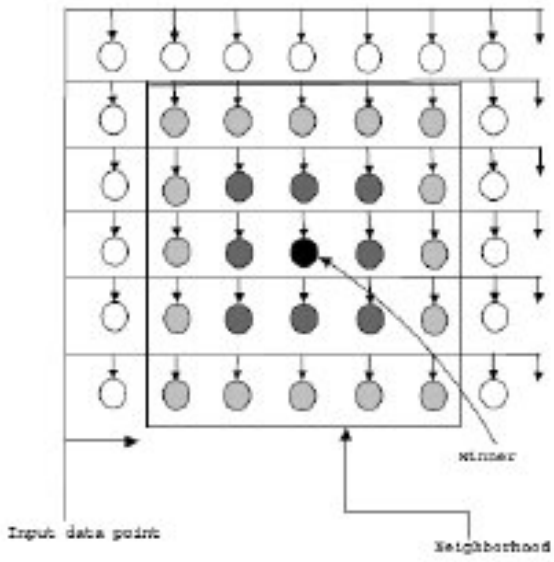


Figure 11: SOM learning scheme.

The way to select the winner is by calculating distances. For its simplicity, the Euclidian distance is used. The winner is the neuron that gives the minimum Euclidean distance. The training function is given by:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)]$$

Where x(t) is the input data point in time t.

m_i is the vector measure of neurons at distance i from the winner

$h_{ci}(t)$ is the neighbourhood function between the winner (c) and the neuron in the i position. The Gaussian function is a common neighbourhood function. The adjustment factor depends on the shape of the Gaussian bell, so the winner is the most adjusted and the neighbour farer the less.

$$h_{ci}(t) = \alpha(t)\exp\left(-\frac{\|r_l, r_j\|^2}{2\sigma^2(l)}\right)$$

Where s(t) is the neighbourhood radio and α is the learning rate factor, if it's big the algorithm goes quickly and if it's small the adjustment is finer. [4]

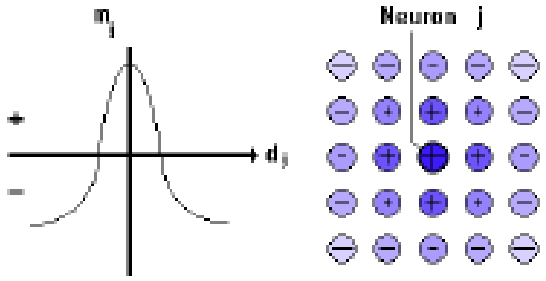


Figure 12: Gaussian neighbourhood function

4. Decision

After the training, the profiles of normal network traffic are built so now it's the time to use the classification to decide if traffic is normal or anomalous. The data set used contained only normal traffic, the neurons represent the profile of normal traffic, if a new input comes and it's far from them, the new input will be detected as an attack.

This algorithm has been shown to be good in detecting attacks [12] but as it has been said it has the disadvantage of the need of a suitable training set.

c. Probabilistic method: SPADE

Another way to make the detection adaptive is by the use probabilistic methods. This kind of algorithms calculates probabilities that a kind of traffic is "normal" or not observing past behaviour of the network. One algorithm that does that is SPADE. SPADE is a pre-processor plug-in for the Snort intrusion detection engine. It detects network traffic that deviates from the "normal" behaviour of the network. Normal traffic depends on each network, for example for a web server the normal traffic is from a large range of external IP addresses to a single address on port 80 TCP. Internal DNS server expects traffic to and from port 53 UDP and TCP from the machines within your network. This can be characterised as "normal" traffic. It would be "not normal" for the internal DNS server to be receiving requests from external IP addresses to port 80 TCP. A signature IDS, as Peakflow, can't detect this kind of anomalies if a signature didn't exist to match. SPADE allows the detection of things that haven't prewritten signatures.

1. Probability tables

In order to detect anomalous packets SPADE maintains probability tables that contain information regarding the number of occurrences of different kinds of packet over time on the network. It assigns a higher weight to more recent occurrences and gradually phases out older occurrences. If P(X) is the probably for a packet X, the “raw anomaly score” a(X) is equal to –log2 (P(X)).

We would know, for example, that P(dip=10.10.10.10,dport=8080) (dip: destination IP, dport: destination port) is 10% but that P(dip=10.10.10.10,dport=8079) is 0.1%. So the anomaly score for a 10.10.10.10, 8080 packet is 3.32 (not very anomalous) and the score for a 10.10.10.10, 8079 packet is 9.97 (fairly anomalous). This result is then normalized by dividing by the highest possible raw anomaly score, so it produces a result which is always between 0 and 1 and thus means the degree of anomalousness in the score is easier to interpret. 0 will represent completely “normal” and 1 completely “not normal’.

In the configuration of SPADE a threshold is set so it will alarm when the score will exceed it.

2. Threshold. Adaptation

In Spade, it is possible to either specify a fixed threshold or let one of the adapt modes adapt it. These modes try to maintain the threshold in a position where the least attacks are missed and the least false alarms are generated. There exists four different ways of doing that but the one that has been proved to be the best in almost all the networks is the method 3. It consists of the average of the ideal threshold values from the last N observations. This mode is invoked with a line of the form:

Pre-processor spade-adapt3: <number of alerts wanted> <number of minutes in a period of observation> <number of observations to average over>

3. Output

Spade produces two types of messages, which, depending on how Spade is configured, are sent to Snort's configured alert or log facilities (e.g., alert file, database, etc.).

The more common one has a message in the form "Spade: <activity description>: <scope>: <anomaly score>", where <activity description> describes what Spade is reporting on, <scope> explains the type of packet that was being examined, and <anomaly score> is the raw or relative anomaly score that Spade has assessed for the packet.

Spade may also periodically produce messages of the form: "Spade: id=<id>: Threshold adjusted to T after X alerts (of N)". This indicates that the alerting threshold for detector <id> was changed to T. This happens when you are using one of the threshold adapting mechanisms (see the Usage file). The message also gives information about the number of alerts (X) sent since the last time the threshold was adjusted and the total number of packets (N) accepted by Spade during that time.

d. Algorithm finally used

The two algorithms presented are known of being good in the task of attack detection. The code of both is available but SOM in Matlab is ready to use and with SPADE in C it is necessary to make some adjustments that we will see later. Apart from that the final decision was to use SPADE for various reasons:

- Speed: C code is quicker than Matlab.
- Simplicity: It integrates the packet's catcher and processing. With the SOM algorithm it is needed to capture the data and extract the interesting features before.
- Generalization: The efficiency of SOM depends on the initial set for training.
- Adaptation: SPADE threshold is adapted on line while SOM's weights are adapted in the training period.

IV. Installation and get off the ground.

SPADE is a pre-processor of Snort so the first thing to do is to install Snort. In this case, I've tried to install some different versions of it and the one that gave fewer problems was the release 1.8.4 that can be downloaded in:

http://gd.tuwien.ac.at/infosys/security/oldsnort/downloads.html

The steps to install Snort are the following:

1. Download the program.
2. Unpack it with tar zxvf snort-stable-snapshot.tar.gz
3. ./configure
4. make

The next thing to do will be to install SPADE but first of all Snort must be tested. For that I make a simple call to the program:

```
snort -c snort.conf -v  -i eth0  -l ./log -A fast
```

With this command Snort is running in mode verbose as a sniffer. The results are also saved in the log file. Connected to the network of the RUS laboratory, the packet exchange is shown below:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/22-10:46:44.301033 129.69.3.202:137 -> 129.69.3.255:137
UDP TTL:128 TOS:0x0 ID:20936 IpLen:20 DgmLen:78
Len: 58
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/22-10:46:45.051052 129.69.3.202:137 -> 129.69.3.255:137
UDP TTL:128 TOS:0x0 ID:21042 IpLen:20 DgmLen:78
Len: 58
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/22-10:46:45.484709 ARP who-has 129.69.3.30 tell 129.69.3.202

08/22-10:46:45.504723 ARP reply 129.69.3.30 is-at 0:0:C:7:AC:1

08/22-10:46:45.824402 129.69.3.202:137 -> 129.69.3.255:137
UDP TTL:128 TOS:0x0 ID:21154 IpLen:20 DgmLen:78
Len: 58
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/22-10:46:46.522920 129.69.3.251:1985 -> 224.0.0.2:1985
UDP TTL:2 TOS:0xC0 ID:0 IpLen:20 DgmLen:48
Len: 28
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Snort is working properly so we're prepared for the installation of SPADE. This step is not so easy since the code has some "errata" for my operating system (Ubuntu 4.0) as source codes not included (commented). The steps needed to install SPADE are:

1. Download the program: release version 092200.1
2. Unpack it with tar zxvf Spade-092200.1.tar.gz
3. Copy spp_anomsensor.c and spp_anomsensor.h to the directory with the snort source code (e.g., the directory with snort.c)
4. Add the line #include "spp_anomsensor.h" to plugbase.h
5. Edit plugbase.c and add "SetupSpade();" to the InitPreprocessors() function
6. Edit Makefile.am and add spp_anomsensor.c and spp_anomsensor.h to the snort_SOURCES line.
7. Recompile snort (e.g., run make)

Let's check now if it works. We call Snort but with the new configuration file. The new output has to have alarms type SPADE. If we choose a very low and fixed threshold and we surf in internet we can see these alarms:

```
[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 1.1443
[**]
08/22-11:42:57.185325 127.0.0.1:1433 -> 129.69.3.234:25
TCP TTL:100 TOS:0x0 ID:50475 IpLen:20 DgmLen:40
******S* Seq: 0x53542E53  Ack: 0xCF67E3B  Win: 0x200  TcpLen: 20

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 0.7444
[**]
08/22-11:42:57.185470 127.0.0.1:1434 -> 129.69.3.234:25
TCP TTL:100 TOS:0x0 ID:13064 IpLen:20 DgmLen:40
******S* Seq: 0x3F072FBC  Ack: 0x320B7564  Win: 0x200  TcpLen: 20
```

As we can see, there are SPADE outputs so the changed program is working.

V. Implementation

a. Place the IDS

The first thing to deal is to choose the place where the IDS should be in the network topology. This depends upon the topology of the network but also upon the intrusion activities we want to detect: internal, external or both. For example, for internal attacks it should be necessary to place an IDS in all the sensitive network segments. For external attacks, the best place to place the IDS is just inside the router connecting to Internet or firewall. Since the more intrusion detection systems mean more work and more maintenance costs, it's important to choose well the place. The decision really depends upon your security policy, which defines what you really want to protect from hackers. The next figure shows typical locations where the IDS can be placed.

.

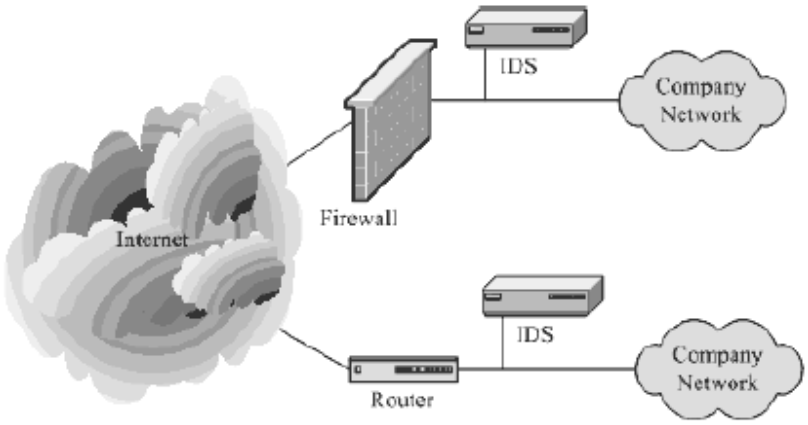


Figure 13: Typical locations for an intrusion detection system.

One important thing to take into account is the use of switches or hubs inside the network. In Ethernet networks there are no problems with hubs because all the ports are connected and we can place the IDS it doesn't matter which port however with a switch is not so easy because the ports are independent. Depending upon the type of switches used, you can use Snort on a switch port. Some switches, like Cisco, allow you to replicate all ports traffic on one port where you can attack the Snort machine. In this case all incoming and outgoing traffic is visible to the IDS.

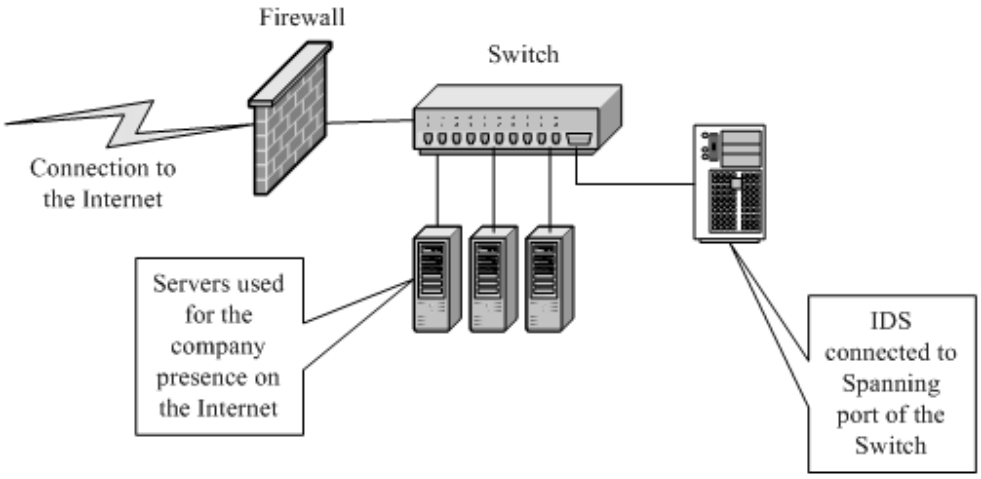


Figure 14: IDS connected to a spanning port.

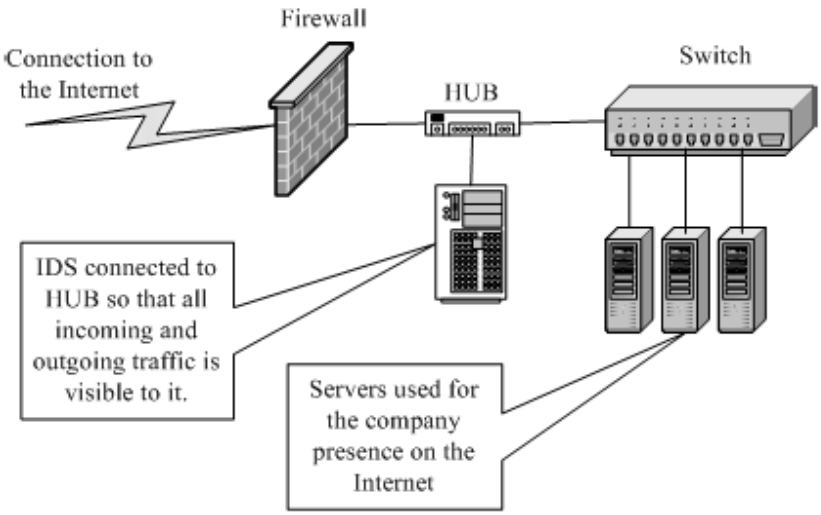


Figure 15: IDS connected to a hub.

It is important to know the network architecture in order to place the IDS in the correct place and do the experiments correctly. For that, I use Ethereal and hear to the interface connected to Internet. I saw connections of all the computers in the network so we can assume that the network is connected with hubs.

The scenario used for the experiments consists of three computers (two for the attacks and one where is the IDS), a hub and a connection to Internet. The diagram is the next:

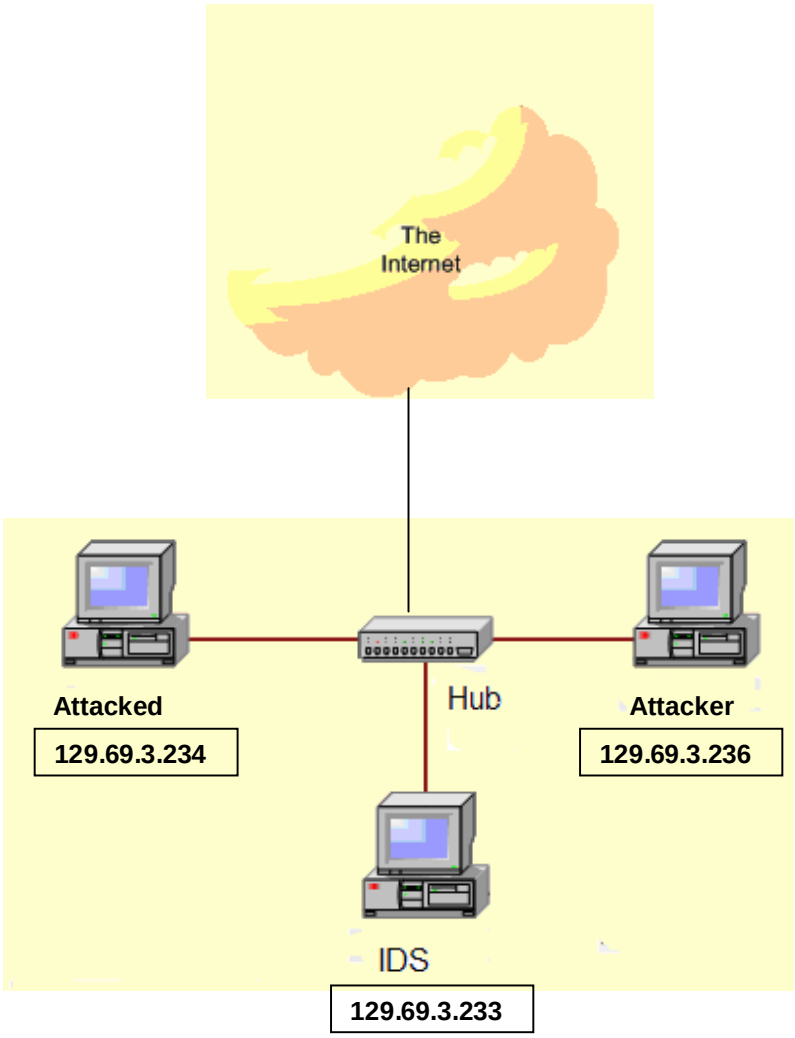


Figure 16: Scenario.

We have to be cleared about the scenario is for tests no real. It is only composed of three computers and the alarms will be generated manually. What it would be interesting is to place the IDS in the same places where Peakflow collects data in order to compare them but it wasn't possible.

b. Simulation of the attacks

As, it has been said before, the attacks have been manually simulated. The tools used are explained as follows.

1. IDSwakeup

To simulate a big amount of attacks in a short period, I choose to use the software IDSwakeup. IDSwakeup is a collection of tools that allows testing network intrusion detection systems. The main goal of IDSwakeup is to generate false attack that mimic well known ones, in order to see if NIDS detects them and generates alarms.

Usage: ./IDSwakeup <src_addr> <dst_addr> [nb] [ttl]

Below is the capture of the attack, as we can see, this program uses typical attacks like:

- teardrop
- ping of death
- MDAC (HTTP POST msadc/msadcs.dll is used to gain access to a system using MDAC (Microsoft Data Access Components) vulnerability).
- Tiny-fragmented TCP
- …

```
# ./IDSwakeup  any 129.69.3.234  1  1

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
-  IDSwakeup : false positive generator
-  Stephane Aubert
-  Hervé Schauer Consultants (c) 2000
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-


  src_addr:0  dst_addr:127.0.0.1  nb:1  ttl:1

  sending : teardrop ...
  sending : land ...
  sending : get_phf ...
  sending : bind_version ...
  sending : get_phf_syn_ack_get ...
  sending : ping_of_death ...
  sending : syndrop ...
  sending : newtear ...
  sending : X11 ...
  sending : SMBnegprot ...
  sending : smtp_expn_root ...
  sending : finger_redirect ...
  sending : ftp_cwd_root ...
  sending : ftp_port ...
  sending : trin00_pong ...
  sending : back_orifice ...
  sending : msadcs ...
            245.146.219.144 -> 127.0.0.1 80/tcp  GET /msadc/msadcs.dll HTTP/1.0
  sending : www_frag ...
            225.158.207.188 -> 127.0.0.1 80/fragmented-tcp
              GET /.............................. HTTP/1.0
               189.210.139.176 -> 127.0.0.1 19/udp  hello
.
.
.
```

2. NMAP

Another tool used in this study is Nmap. Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. It scans networks to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. With Nmap we'll be able to simulate network scan attacks.

It can be downloaded in: http://nmap.org/ and executed: nmap –options <ip_address>

```
> sudo nmap -sS 129.69.3.234

Starting Nmap 4.53 ( http://insecure.org ) at 2008-09-25 14:01
CEST

All 1714 scanned ports on dyn-3-14.rus.uni-stuttgart.de
(129.69.3.234) are closed

MAC Address: 00:01:03:8C:B9:E9 (3com)

Nmap done: 1 IP address (1 host up) scanned in 0.745 seconds
```

The option that we will used is the above one –sS, that means Stealth Scan. Stealth scanning sends a SYN packet and looks at the response. If SYN/ACK is sent back, the port is open and the remote end is trying to open a TCP connection. The scanner then sends an RST to tear down the connection before it can be established fully; often preventing the connection attempt appearing in application logs. If the port is closed, an RST will be sent. If it is filtered, the SYN packet will have been dropped and no response will be sent. In this way, Nmap can detect three port states - open, closed and filtered.

3. Spike

Spike.sh is a tool that allows executing much kind of DoS attacks to a host. It can be downloading in:

http://euitio178.ccu.uniovi.es/wiki/index.php/Ataques_de_denegaci%C3%B3n_de_servicio_(DoS)#Herramientas_para_realizar_pruebas_de_ataques_DoS

And executed in the Linux console by: sh spike.sh <ip_address> and then the options must be chosen from the menus.

```
> sh spike.sh 129.69.3.234

############## Main Menu For spike.sh ##############

# 1] Combo Attacks #

# 2] Pick The Attack #

# 3] Pick The Attack2 (Enter Your Own Info) #

# 4] Class C Attack (Drop A Class C) #

# 5] Never Die (Attacks That Never Die) #

# 6] Clean up (Kill Hung Attacks) #

# 7] Info Menu (Other Info About spike.sh) #

# 8] To Exit Out #

Enter The Number Of The Option To Run

#> 1

######## ComboAttack Menu ########

# 1] Light Attack (14 Attacks) #

# 2] Medium Attack (22 Attacks) #

# 3] Hard Attack (36 Attacks) #

# 4] To Exit Out #

Enter The Number Of The Option To Run

#> 1

Teadroping.....

Completed....

Kocing.....

Completed....

Targa3ing......

Completed....

Pimping.....

.

.

.
```

VI. Experiments

For the experiments, the next lines have been included in the configuration file:

```
var SPADEDIR /var/log/spade

preprocessor spade: 15 $SPADEDIR/spade.rcv $SPADEDIR/log.txt 3 50000
preprocessor spade-homenet: 129.69.3.0/24
preprocessor spade-adapt3: 0.01 60 168
preprocessor spade-survey:  $SPADEDIR/survey.txt 3
```

That is to say that the log file is going to be saved in the directory /var/log/spade, the threshold is initially 15, the net to be analysed is 129.69.3.0/24, which is, as indicated previously, our test network, the algorithm used to adapt the threshold is the third, that was explained earlier. The survey mode reports information periodically about the anomaly scores produced in the last time interval (3 minutes) in the file survey.txt.

a. First experiment: attack detection from Internet

The first think done to test the IDS is to run it looking into the Internet network without any work of the computers of our scenario. It has been kept for 18 hours and the output is the following:

```
===============================================================================
Snort analyzed 112496 out of 563658 packets, The kernel dropped 0(0.000%) packets

Breakdown by protocol:                Action Stats:
    TCP: 11889      (2.109%)          ALERTS: 0
    UDP: 50783      (9.010%)          LOGGED: 0
   ICMP: 185        (0.033%)          PASSED: 0
    ARP: 10643      (1.888%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 38996      (6.918%)
DISCARD: 0          (0.000%)
===============================================================================
Fragmentation Stats:
Fragmented IP Packets: 0          (0.000%)
    Fragment Trackers: 0
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
===============================================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 11837      (2.100%)
         Stream Trackers: 985
          Stream flushes: 9
           Segments used: 17
   Stream4 Memory Faults: 0
===============================================================================
Snort received signal 2, exiting
```

The output shows the number and the type of the packets analysed, as well as the number of alerts that they cause. As it can be seen, no alarms have been occurred during this time.

b. Second experiment: IDSwakeup attack between hosts

The next step is to cause the attacks in order to prove the efficacy of our IDS. The first kind of attacks has been done with the IDSwakeup, so a lot of different attacks have been induced from the attacker to the attacked.

And the result is the following:

```
===============================================================================
Snort analyzed 2829 out of 9675 packets, The kernel dropped 1407(14.543%) packets

Breakdown by protocol:                Action Stats:
    TCP: 1685       (17.416%)         ALERTS: 462
    UDP: 768        (7.938%)          LOGGED: 462
   ICMP: 2          (0.021%)          PASSED: 0
    ARP: 95         (0.982%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 260        (2.687%)
DISCARD: 18         (0.186%)
===============================================================================
Fragmentation Stats:
Fragmented IP Packets: 103        (1.065%)
    Fragment Trackers: 2
   Rebuilt IP Packets: 1
   Frag elements used: 2
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
===============================================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 1342       (13.871%)
         Stream Trackers: 134
          Stream flushes: 11
           Segments used: 27
   Stream4 Memory Faults: 0
===============================================================================
Snort received signal 2, exiting
```

Now a lot of alarms have been generated. Let's take a look on the alert's log to see what kind of alerts are.

```
[**] [1:269:1] DOS Land attack [**]
[Classification: Attempted Denial of Service] [Priority: 2]
08/22-11:11:30.248661 129.69.3.234:53 -> 129.69.3.234:53
TCP TTL:100 TOS:0x0 ID:3868 IpLen:20 DgmLen:40
******S* Seq: 0xF1C  Ack: 0x0  Win: 0x800  TcpLen: 20

[**] [1:269:1] DOS Land attack [**]
[Classification: Attempted Denial of Service] [Priority: 2]
08/22-11:11:30.248693 129.69.3.234:53 -> 129.69.3.234:53
TCP TTL:100 TOS:0x0 ID:3868 IpLen:20 DgmLen:40
******S* Seq: 0xF1C  Ack: 0x0  Win: 0x800  TcpLen: 20

[**] [1:269:1] DOS Land attack [**]
[Classification: Attempted Denial of Service] [Priority: 2]
08/22-11:11:30.248730 129.69.3.234:53 -> 129.69.3.234:53
TCP TTL:100 TOS:0x0 ID:3868 IpLen:20 DgmLen:40
******S* Seq: 0xF1C  Ack: 0x0  Win: 0x800  TcpLen: 20
.
.
.
[**] [1:257:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/22-11:11:30.256559 255.255.255.255:1249 -> 129.69.3.234:53
UDP TTL:100 TOS:0x0 ID:9018 IpLen:20 DgmLen:58
Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

.
.
.
```

The alerts show the type of alert, the time that has been generated and the heather of the packet. As it has been explained before, these aren't SPADE alerts. All the alarms reported are Snort ones, no one from Spade. The first attempt to solve that is to change the initial threshold to a lower one, for example 2:

```
preprocessor spade: 2 $SPADEDIR/spade.rcv $SPADEDIR/log.txt 3 50000
```

But the results were not as expected. We obtained a few Spade alarms but not all as it should.

```
[**] [1:257:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/22-11:42:49.909617 255.255.255.255:1249 -> 129.69.3.234:53
UDP TTL:100 TOS:0x0 ID:9018 IpLen:20 DgmLen:58
Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:257:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/22-11:42:49.909639 255.255.255.255:1249 -> 129.69.3.234:53
UDP TTL:100 TOS:0x0 ID:9018 IpLen:20 DgmLen:58
Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 4.2486
[**]
08/22-11:42:49.911988 255.255.255.255:1634 -> 129.69.3.234:80
TCP TTL:100 TOS:0x0 ID:8879 IpLen:20 DgmLen:60 DF
******S* Seq: 0xED7549E4  Ack: 0x0  Win: 0x7D78  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 15499042 0 NOP WS: 0
```

The supposition therefore is that as Spade is a non usual traffic detector and the IDSwakeup make continuous attacks, Spade can treat these attacks as normal traffic. So, the purpose for the next experiment is to make single attacks.

c. Third experiment: Nmap attack between hosts

To do that, the Nmap tool is executed and the results are shown bellow:

```
===============================================================================
Snort analyzed 761 out of 4521 packets, The kernel dropped 2961(65.494%)
packets

Breakdown by protocol:                Action Stats:
    TCP: 743        (16.434%)         ALERTS: 181
    UDP: 7          (0.155%)          LOGGED: 2
   ICMP: 0          (0.000%)          PASSED: 0
    ARP: 6          (0.133%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 5          (0.111%)
DISCARD: 0          (0.000%)
===============================================================================
Fragmentation Stats:
Fragmented IP Packets: 0          (0.000%)
    Fragment Trackers: 0
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
===============================================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 743        (16.434%)
         Stream Trackers: 268
          Stream flushes: 59
           Segments used: 118
   Stream4 Memory Faults: 0
===============================================================================
Snort received signal 2, exiting
```

```
[**] [100:1:1] spp_portscan: PORTSCAN DETECTED from 129.69.3.236 (THRESHOLD 4
connections exceeded in 0 seconds) [**]
08/22-17:32:19.323355

[**] [1:615:2] SCAN Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/22-17:32:19.325278 129.69.3.236:39668 -> 129.69.3.234:1080
TCP TTL:59 TOS:0x0 ID:31009 IpLen:20 DgmLen:44
******S* Seq: 0x7705D703  Ack: 0x0  Win: 0x1000  TcpLen: 24
TCP Options (1) => MSS: 1460
[Xref => http://help.undernet.org/proxyscan/]

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 4.0029 [**]
08/22-17:32:19.328949 129.69.3.236:39668 -> 129.69.3.234:66
TCP TTL:49 TOS:0x0 ID:62207 IpLen:20 DgmLen:44
******S* Seq: 0x7705D703  Ack: 0x0  Win: 0x800  TcpLen: 24
TCP Options (1) => MSS: 1460

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 4.0155 [**]
08/22-17:32:19.328991 129.69.3.236:39668 -> 129.69.3.234:2809
TCP TTL:51 TOS:0x0 ID:44746 IpLen:20 DgmLen:44
******S* Seq: 0x7705D703  Ack: 0x0  Win: 0x1000  TcpLen: 24
TCP Options (1) => MSS: 1460

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 4.0279 [**]
08/22-17:32:19.329047 129.69.3.236:39668 -> 129.69.3.234:5400
TCP TTL:49 TOS:0x0 ID:11838 IpLen:20 DgmLen:44
******S* Seq: 0x7705D703  Ack: 0x0  Win: 0x800  TcpLen: 24
TCP Options (1) => MSS: 1460

.
.
.
```

In that case, Spade is detecting the anomalies. As we can see, Snort can label this attack too so we comment the corresponding rule in order that we see only Spade working.

```
[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 2.3312 [**]
08/22-11:42:57.208695 127.0.0.1:1444 -> 129.69.3.234:25
TCP TTL:100 TOS:0x0 ID:60756 IpLen:20 DgmLen:40
******S* Seq: 0x79294530  Ack: 0x7313F971  Win: 0x200  TcpLen: 20

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 2.2623 [**]
08/22-11:42:57.214259 127.0.0.1:1445 -> 129.69.3.234:25
TCP TTL:100 TOS:0x0 ID:44666 IpLen:20 DgmLen:40
******S* Seq: 0x1E87839  Ack: 0x1828E6E1  Win: 0x200  TcpLen: 20

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 2.1984 [**]
08/22-11:42:57.214305 127.0.0.1:1446 -> 129.69.3.234:25
TCP TTL:100 TOS:0x0 ID:41584 IpLen:20 DgmLen:40
******S* Seq: 0x6453DAF5  Ack: 0x6066824C  Win: 0x200  TcpLen: 20

.
.
.
```

In that case it works because SPADE builds the probabilities table based on destination IP and port, since attacks are against different ports, it considers them different attacks.

d. Fourth experiment: Spike attack between hosts

The last experiment is to test the IDS with the most common attack: Denial of Service (DOS). For that, we use Spike.

```
===============================================================================
Snort analyzed 5510 out of 11659 packets, The kernel dropped 53(0.455%) packets

Breakdown by protocol:                Action Stats:
    TCP: 2909       (24.951%)         ALERTS: 0
    UDP: 59         (0.506%)          LOGGED: 0
   ICMP: 81         (0.695%)          PASSED: 0
    ARP: 22         (0.189%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 45         (0.386%)
DISCARD: 0          (0.000%)
===============================================================================
Fragmentation Stats:
Fragmented IP Packets: 2532       (21.717%)
    Fragment Trackers: 60
   Rebuilt IP Packets: 57
   Frag elements used: 2451
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
===============================================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 2902       (24.891%)
         Stream Trackers: 7
          Stream flushes: 0
           Segments used: 0
   Stream4 Memory Faults: 0
===============================================================================
Snort received signal 2, exiting
```

No alarms have been detected. So we uncommented the rules of DOS in the snort configuration file to see if Snort detects them.

```
===============================================================================
Snort analyzed 5886 out of 12152 packets, The kernel dropped 0(0.000%) packets

Breakdown by protocol:                Action Stats:
    TCP: 3162       (26.020%)         ALERTS: 62
    UDP: 61         (0.502%)          LOGGED: 62
   ICMP: 85         (0.699%)          PASSED: 0
    ARP: 14         (0.115%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 44         (0.362%)
DISCARD: 0          (0.000%)
===============================================================================
Fragmentation Stats:
Fragmented IP Packets: 2580       (21.231%)
    Fragment Trackers: 60
   Rebuilt IP Packets: 60
   Frag elements used: 2580
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
===============================================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 3152       (25.938%)
         Stream Trackers: 9
          Stream flushes: 769
           Segments used: 1536
   Stream4 Memory Faults: 0
===============================================================================
Snort received signal 2, exiting
```

```
[**] [1:274:1] DOS ath [**]
[Classification: Attempted Denial of Service] [Priority: 2]
08/22-18:48:07.168525 129.69.3.236 -> 129.69.3.234
ICMP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:31527   Seq:256  ECHO
[Xref => http://www.whitehats.com/info/IDS264]

[**] [1:274:1] DOS ath [**]
[Classification: Attempted Denial of Service] [Priority: 2]
08/22-18:48:08.167594 129.69.3.236 -> 129.69.3.234
ICMP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:31527   Seq:512  ECHO
[Xref => http://www.whitehats.com/info/IDS264]

[**] [1:480:2] ICMP PING speedera [**]
[Classification: Misc activity] [Priority: 3]
08/22-18:48:08.177811 129.69.3.236 -> 129.69.3.234
ICMP TTL:64 TOS:0x0 ID:28449 IpLen:20 DgmLen:63228
Type:8  Code:0  ID:31528   Seq:256  ECHO

[**] [1:499:1] MISC Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
08/22-18:48:08.183735 129.69.3.234 -> 129.69.3.236
ICMP TTL:64 TOS:0x0 ID:11705 IpLen:20 DgmLen:63228
Type:0  Code:0  ID:31528  Seq:256  ECHO REPLY
[Xref => http://www.whitehats.com/info/IDS246]
```

Snort detects properly the DoS attack because it has the rules defined. However SPADE can't detect them because of its probabilistic feature. This attack is very repetitive so SPADE considers it normal traffic

e. Fifth experiment

One of the most important requirements for IDSs is that they should be effective, that is, detect a substantial percentage of intrusions, while keeping the false-positive rate at an acceptable level. With the previous experiments, it wasn't possible to check the false-positives so this experiment will consist in checking this feature. To know the false alarms it is necessary to have a known traffic data-base, in this case the 1999 DARPA/MIT Lincoln Laboratory off-line intrusion detection evaluation data set, weeks 4 and 5[13]. It is known that during these weeks the number of attacks is 201 instances of 58 different attacks.

Passing the DARPA data set to SPADE, we obtained 20136 alarms (threshold exceeded) but we have to take in consideration that DARPA informs that duplicate attacks with less of 60 seconds of leeway are the same attack and have to be ignored. After deleting the duplicated alarms, the number of total alarms detected by SPADE is 5162.

So, the percentage of false positives is very high. SPADE is based on the average frequency in training, as there is no specific training period; all packets are added to the training model after evaluation and if there isn't a big period of "normal" traffic, SPADE can't build the probability table correctly. SPADE isn't good with DARPA data set because the frequency of the packets isn't high. The solution will be a very huge data set with a long period of non attack first. This way, SPADE would be able to build the probabilities accordingly.

Let check now, how the efficacy varies with the threshold. For that, it is common to use the ROC (Receiver Operating Characteristic) curve, which allows checking the relation between the true positive and the false positive rate when a parameter is changing, in this case the threshold. For that, the line for the adaptation of the threshold is commented and the threshold is changed manually between 1 and 15. The data set used has been only the last day of the last week. The ROC curve is drawn in Matlab.

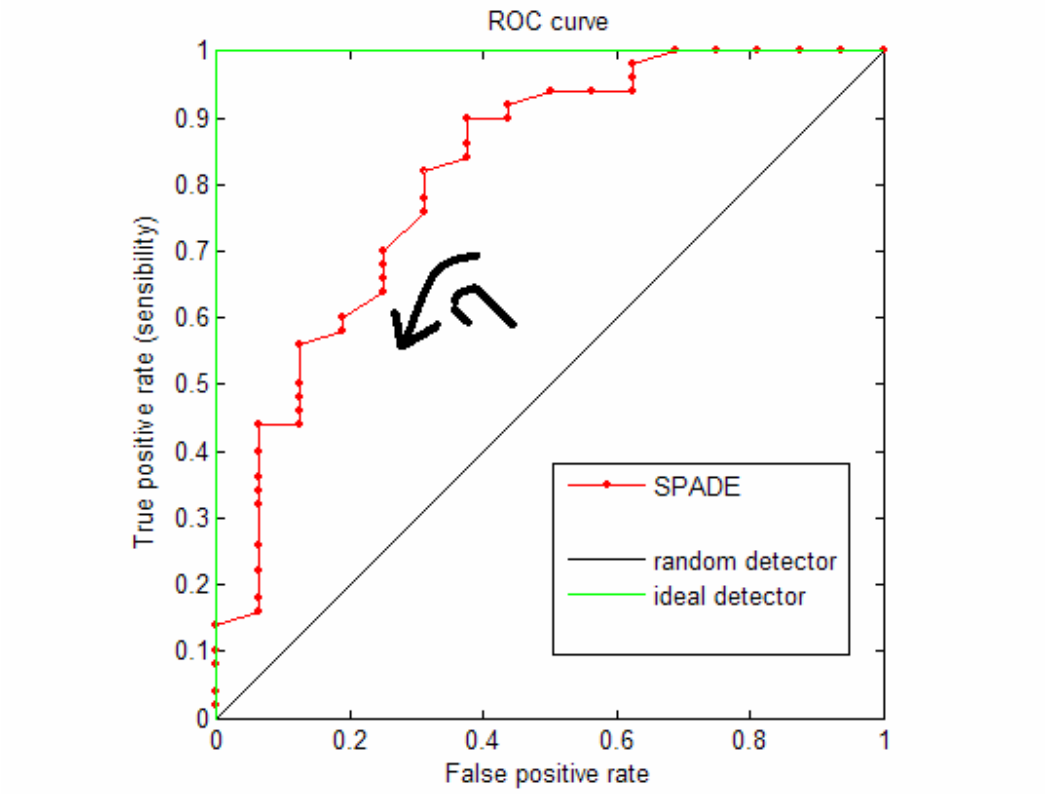


Figure 17: ROC curve of SPADE.

It can be seen in the figure that the rates increase when the threshold decreases, this is logical; since the smallest is the threshold the most of the traffic overpass it. A detector quality is measured in terms of being able to detect as most of attacks possible at the same time that it detects the least false positives. There is a compromise between the sensibility of the detector and the low false positive rate.

f. Conclusions of the experiments

From the experiments, it can be concluded that each IDS has its defects. Misuse IDS have the problem of no detecting novel attacks but the anomaly based IDS introduce many false alarms. If the sensibility of the detector is high, the number of false alarms will be important and the administrator won't trust on it. However the pursuit of detecting as many intrusions as possible produces too many false positives, building an effective IDS that generates only a small number of false positives is an extremely difficult task.

Another problem using anomaly-based IDS is that they don't recognize continuous attacks as they consider them as "normal" traffic when they take a long time.

VII. Problems found

The first to deal was the lack of material (pcs working and installation of linux). The installation of the tools and the different libraries to make them work in linux was complicated. Find a data set and find the false alarms in the SPADE generated file have taken a lot of time.

VIII. Conclusions and further work

In response to new potential attacks, IDS are improving continuously. Here two techniques based on an anomaly-detection approach have been presented in order to ameliorate the rate detection attack of the University of Stuttgart.

The neural network detector, SOM, hasn't been chosen because of the need of extracting the good features to analyse and because of the training period that has to be executed with a large and good chosen data set, that is to say, that permits generalization.

SPADE detects attacks probabilistically, allowing for generalization. However, they lack attack models and can potentially "learn" to consider an attack normal and adapt itself along the time. His majors' limitations are the possible high false alarm rate and the need of a period of normal traffic in order to build the probabilities table precisely. A future work could be to make a better characterization of normal traffic.

Therefore, SPADE hasn't been proved to be a good IDS on its own but it can be a good complement of a misuse one. Anyway, for the moment it seems necessary in some cases the action of a human expert for the last decision.

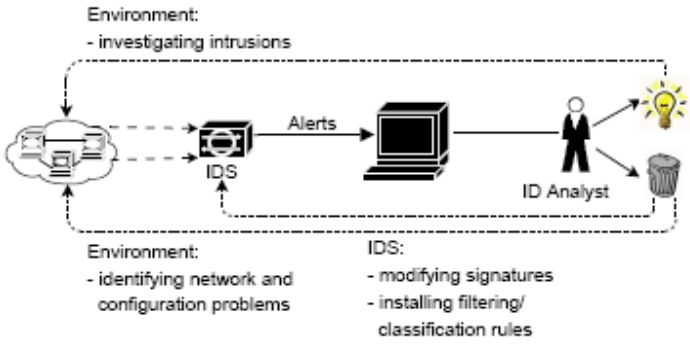


Figure 18: Alert management scenario

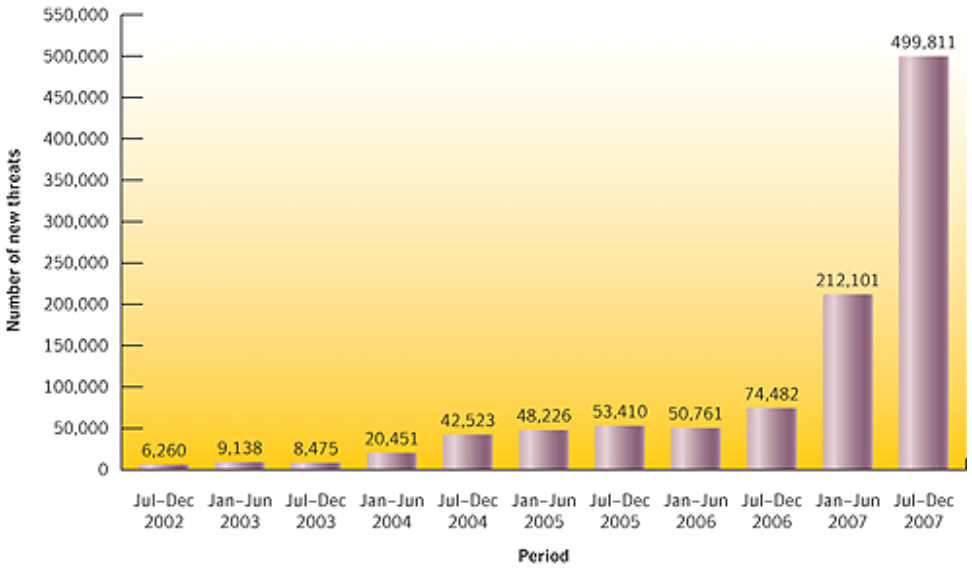
IX. Resumen en español

a. Introducción y puesta en marcha

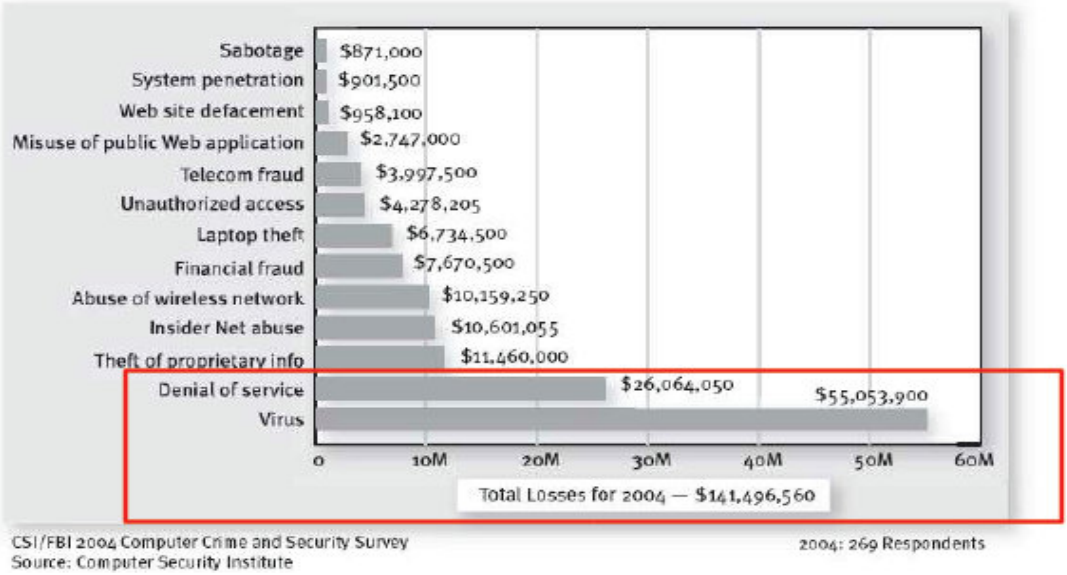
La proliferación de Internet y el aumento del número de redes de ordenadores están provocando un incremento de ataques a la red que atentan a diferentes aspectos de la comunicación:

- Integridad: Fiabilidad de la información.
- Disponibilidad: los recursos tienen que estar disponibles cuando se necesitan.
- Confidencialidad: acceso limitado a la información a usuarios autorizados.

En 2007, Symantec detectó un incremento del 468% en el número de nuevos ataques con respecto al año anterior.



Los ataques afectan significativamente al rendimiento de la red. Un estudio de el Information Sciences Institute de California muestra que, por ejemplo, los ataques DDoS causan aumentos de latencia del 230% para el tráfico DNS y del 30% para el tráfico web [2]. Esto, visto desde un punto de vista financiero, se traduce en un coste elevado. Como se puede ver en la siguiente figura, los virus indujeron un coste de 5 millones de dólares en 2004 [3].



Hay múltiples tipos de ataques a la red y cambian continuamente a medida que los atacantes tienen más conocimiento y los hacen más sofisticados.

El tráfico malicioso puede clasificarse en diferentes grupos, en este proyecto sólo se han considerado tres como los más útiles para los experimentos desarrollados:

- Gusano: programa que tiene la propiedad de duplicarse a si mismo y mandar sus copias a otros nodos.
- Escaneo de puertos: programa que audita redes y máquinas con el fin de saber qué puertos están abiertos y cuáles no y acceder a la máquina a través de ellos.
- Ataques de Denegación de Servicio (DoS) o ataques distribuidos de Denegación de Servicio (DDos): causa que un servicio o recurso sea inaccesible a los usuarios legítimos. Un método común para este tipo de ataques consiste en saturar a la víctima con muchas peticiones de conexión y hacer que se pierda la conectividad por sobrecarga de la red. El ataque puede ser desde una máquina (DoS) o desde múltiples (DDoS). Otros métodos de ataque DoS son: Smurf, Ping of Death y Teardrop.

El tráfico malicioso infecta las redes, es costoso y afecta a la eficiencia de la red; es necesario controlarlo para preservar la disponibilidad de la red. Es por ello, que los Sistemas de Detección de Intrusión (Intrusion Detection System – IDS) se han convertido en una parte esencial de las infraestructuras de seguridad de hoy en día. Existen diferentes tipos de IDS:

- Detección de usos indebidos: se basa en especificar de una forma más o menos formal las potenciales intrusiones que amenazan a un sistema y simplemente esperar a que alguna de ellas ocurra. Es decir, compara las características del tráfico de la red con patrones de ataques comunes. Este tipo de sistema no es útil para intrusiones no conocidas. Como se dijo antes, los tipos de ataques cambian continuamente por lo que los patrones tienen que ser actualizados con la misma periodicidad.

- Detección de anomalías: estos modelos de detección conocen lo que es “normal” en nuestra red o nuestras máquinas a lo largo del tiempo, desarrollando y actualizando conjuntos de patrones contra los que comparar los eventos que se producen en los sistemas. Si uno de esos eventos (por ejemplo, una trama procedente de una máquina desconocida) se sale del conjunto de normalidad, automáticamente se cataloga como sospechoso. Se pueden clasificar a su vez en adaptativos y no-adaptativos. La adaptación se refiere a la capacidad del IDS de modificar alguno de sus parámetros de acuerdo a observaciones pasadas y así seguir los cambios del tráfico “normal” de la red.

En la universidad de Stuttgart, el sistema de monitorización de la red se llama Peakflow y se basa en la detección de usos indebidos a través de patrones por lo que no es eficiente para la detección de nuevos ataques. Por lo tanto, el objetivo de este proyecto consistía en mejorar este sistema proponiendo una detección basada en anomalías.

Se propusieron dos métodos de detección basada en anomalías:

- Modelo de mapas auto-organizados SOM (Self-Organizing Maps): estos mapas presentan la característica de organizar la información de entrada, de entre un gran volumen de datos, clasificándola automáticamente lo que permite visualizar relaciones importantes entre datos. Este modelo es muy útil para establecer relaciones desconocidas previamente. Se basa en el cerebro humano que adquiere conocimiento con el aprendizaje y lo almacena con conexiones interneuronales. Es por ello, que este tipo de modelos tiene que ser entrenado con una gran cantidad de datos correspondientes a tráfico “normal” y así obtener una buena generalización para la clasificación entre tráfico “normal” e infectado.
- Modelo estadístico SPADE: este modelo calcula la probabilidad de que el tráfico actual sea “normal” en base al tráfico anterior. Crea una tabla de probabilidades que contienen la información del número de ocurrencias en la red de tipos de paquetes diferentes a lo largo del tiempo. Asigna un peso mayor a las ocurrencias más recientes y uno gradualmente menor a las ocurrencias más antiguas. El sistema considerará que el tráfico es malicioso cuando la probabilidad rebase cierto umbral. Este umbral es adaptativo y se fija haciendo la media de los umbrales ideales de los casos anteriores.

  SPADE es un preprocesador para Snort. Snort es un IDS basado en reglas estáticas. SPADE es un proyecto creado originalmente por James Hoagland de Silicon Defense, pero que desapareció hace tiempo por lo que no está muy mantenido, pero al menos las fuentes del trabajo en cvs están disponibles. Con un poco de paciencia, y dado que el código de SPADE es bastante modular se puede añadir a la versión actual de Snort.

  Los mensajes generados por SPADE son del tipo:

  Spade: <descripción actividad>: <campo>: <resultado anomalía>

  Donde <descripción actividad> describe sobre qué SPADE está informando
  <campo> explica cuál es el tipo de paquete que se está examinando
  <resultado anomalía> es la puntuación que SPADE ha asignado al paquete.

Periódicamente, SPADE genera otro tipo de mensajes del tipo:

Spade: id=<id>: Threshold adjusted to T after X alerts (of N)

Esto indica que el umbral se ha adaptado después de X alertas y N paquetes.

Los dos algoritmos podrían valer para el estudio pero, finalmente, se decidió usar SPADE por diferentes razones:

- Rapidez: el código está escrito en C que es más rápido que Matlab, que es el que usa SOM.
- Simplicidad: integra el capturador y procesador de paquetes mientras que si se usa SOM habría que capturar los datos y extraer las características interesantes.
- Generalización: la eficiencia de SOM depende del conjunto inicial de entrenamiento.
- Adaptación: el umbral de SPADE se adapta en línea mientras que los pesos de SOM se adaptan en el periodo de entrenamiento.

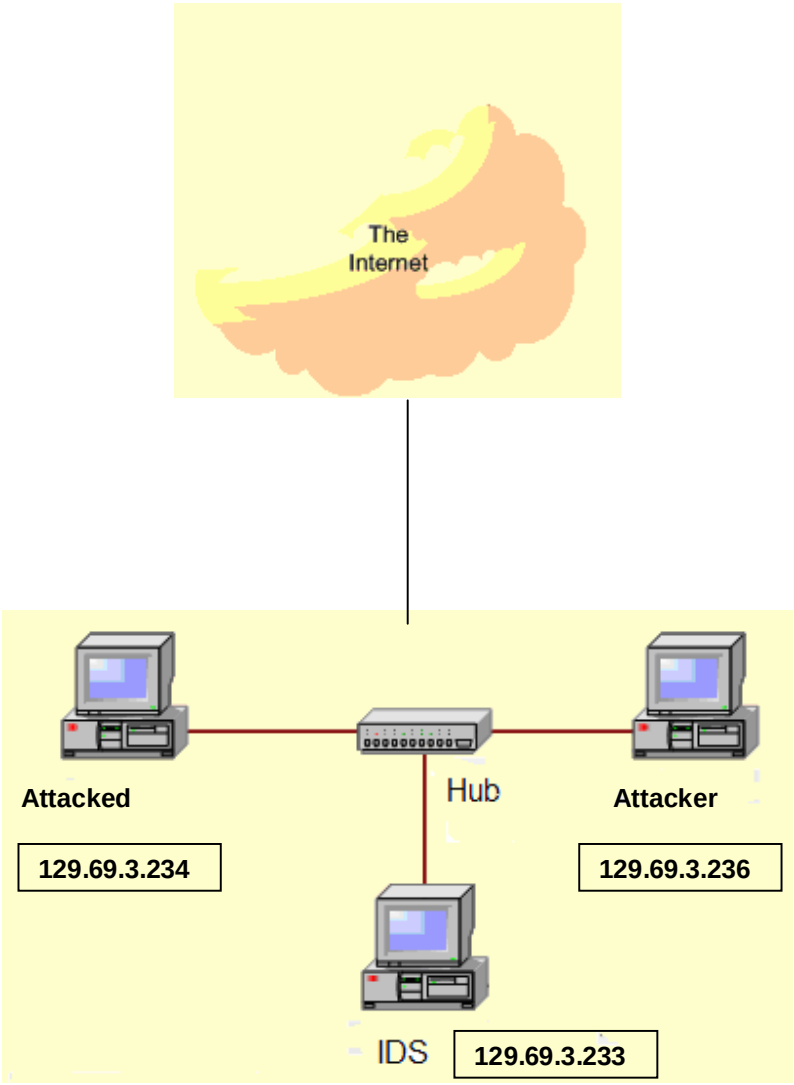
Los pasos seguidos para comenzar con los experimentos fueron los siguientes:

1) Instalación y testeo de SNORT
2) Configuración, instalación y testeo de SPADE
3) Poner en marcha el entorno. Hay varias maneras de colocar el IDS en una red y depende de la topología de la red y del tipo de intrusiones que se quieran detectar: internas, externas o ambas. Por ejemplo, para ataques internos habría que colocar el IDS en todos los segmentos sensibles de la red. En cambio, para ataques externos, el mejor lugar para colocar el IDS sería en el router que conecta a Internet o en el firewall. Es un tema importante ya que los costes dependerán en gran medida de cuántos IDS se coloquen y en qué posición se coloquen para hacerlos más eficientes.

   Otro tema importante a tener en cuenta es el uso de switches o hubs en la red. En redes ethernet no hay problemas con los hubs porque todos los puertos están conectados y se puede conectar el IDS en cualquiera. Sin embargo, con un switch no es tan fácil porque los puertos son independientes. En algunos tipos de switch, como los Cisco, es posible replicar todo el tráfico de los puertos en un puerto (spanning port) donde se colocaría el IDS.

   Por lo tanto, en este paso, fue importante detectar el tipo de red que existía en la universidad. Para ello, se uso Ethereal y se “escuchó” por un interfaz conectado a Internet. Se vieron las conexiones de todos los ordenadores de la red por lo que se asumió que la red estaba conectada con hubs.

   Finalmente el escenario que se utilizó para los experimentos consistió en tres ordenadores (dos para los ataques y uno para el IDS), un hub y una conexión a Internet. El diagrama es el siguiente.



Lo que está claro es que el escenario utilizado para los tests no es real. Está compuesto de tres ordenadores y las alarmas serán generadas manualmente. Lo que hubiera sido interesante hubiese sido colocar los IDS en los mismos lugares donde Peakflow recoge los datos pero, por temas de seguridad, no fue posible.

4) Puesta en marcha de los ataques que se van a utilizar:

- IDSwakeup: simula una gran cantidad de ataques un periodo corto de tiempo. Los ataques que simula son: teardrop, ping of death, MDAC, fragmentos cortos TCP,…
- NMAP ("Network Mapper"): simula ataques de escaneo de la red.
- Spike: simula ataques DoS.

b. Experimentos

Se configuró el algoritmo para que analizara la red 129.69.3.0/24 e informara de las anomalías cada tres minutes en un fichero llamado survey.txt. Se hicieron cinco experimentos:

1. Detección de ataques desde Internet.

Se tuvo el IDS analizando el tráfico proveniente de Internet durante 18 horas pero no se obtuvo ningún ataque en ese tiempo. La salida obtenida fue la siguiente:

```
===============================================================================
Snort analyzed 112496 out of 563658 packets, The kernel dropped 0(0.000%) packets

Breakdown by protocol:                Action Stats:
    TCP: 11889      (2.109%)          ALERTS: 0
    UDP: 50783      (9.010%)          LOGGED: 0
   ICMP: 185        (0.033%)          PASSED: 0
    ARP: 10643      (1.888%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 38996      (6.918%)
DISCARD: 0          (0.000%)
===============================================================================
Fragmentation Stats:
Fragmented IP Packets: 0          (0.000%)
    Fragment Trackers: 0
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
===============================================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 11837      (2.100%)
         Stream Trackers: 985
          Stream flushes: 9
           Segments used: 17
   Stream4 Memory Faults: 0
===============================================================================
Snort received signal 2, exiting
```

Se provocaron múltiples ataques y SPADE no detectó todos ya que al haber muchos ataques en poco tiempo, SPADE considera finalmente que es tráfico normal.

3. Ataques con Nmap.

En este caso, SPADE sí que detecta bien los ataques:

```
[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 2.3312 [**]
08/22-11:42:57.208695 127.0.0.1:1444 -> 129.69.3.234:25
TCP TTL:100 TOS:0x0 ID:60756 IpLen:20 DgmLen:40
******S* Seq: 0x79294530  Ack: 0x7313F971  Win: 0x200  TcpLen: 20

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 2.2623 [**]
08/22-11:42:57.214259 127.0.0.1:1445 -> 129.69.3.234:25
TCP TTL:100 TOS:0x0 ID:44666 IpLen:20 DgmLen:40
******S* Seq: 0x1E87839  Ack: 0x1828E6E1  Win: 0x200  TcpLen: 20

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 2.1984 [**]
08/22-11:42:57.214305 127.0.0.1:1446 -> 129.69.3.234:25
TCP TTL:100 TOS:0x0 ID:41584 IpLen:20 DgmLen:40
******S* Seq: 0x6453DAF5  Ack: 0x6066824C  Win: 0x200  TcpLen: 20

.
.
.
```

Esto se debe a que, como SPADE construye la tabla de probabilidades en base a los puertos e IPs, y los ataques se hacen a puertos diferentes, entonces las alarmas son consideradas diferentes y no pasa como en el caso anterior.

4. Ataque Spike.

En este caso, vuelve a ocurrir como en el segundo experimento. Los ataques DoS son muy repetitivos así que SPADE no los detecta correctamente, lo considera tráfico normal. En cambio Snort, al tener la regla definida, sí que los detecta:

```
[**] [1:274:1] DOS ath [**]
[Classification: Attempted Denial of Service] [Priority: 2]
08/22-18:48:07.168525 129.69.3.236 -> 129.69.3.234
ICMP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:31527   Seq:256  ECHO
[Xref => http://www.whitehats.com/info/IDS264]

[**] [1:274:1] DOS ath [**]
[Classification: Attempted Denial of Service] [Priority: 2]
08/22-18:48:08.167594 129.69.3.236 -> 129.69.3.234
ICMP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:31527   Seq:512  ECHO
[Xref => http://www.whitehats.com/info/IDS264]

[**] [1:480:2] ICMP PING speedera [**]
[Classification: Misc activity] [Priority: 3]
08/22-18:48:08.177811 129.69.3.236 -> 129.69.3.234
ICMP TTL:64 TOS:0x0 ID:28449 IpLen:20 DgmLen:63228
Type:8  Code:0  ID:31528   Seq:256  ECHO

[**] [1:499:1] MISC Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
08/22-18:48:08.183735 129.69.3.234 -> 129.69.3.236
ICMP TTL:64 TOS:0x0 ID:11705 IpLen:20 DgmLen:63228
Type:0  Code:0  ID:31528  Seq:256  ECHO REPLY
[Xref => http://www.whitehats.com/info/IDS246]
```

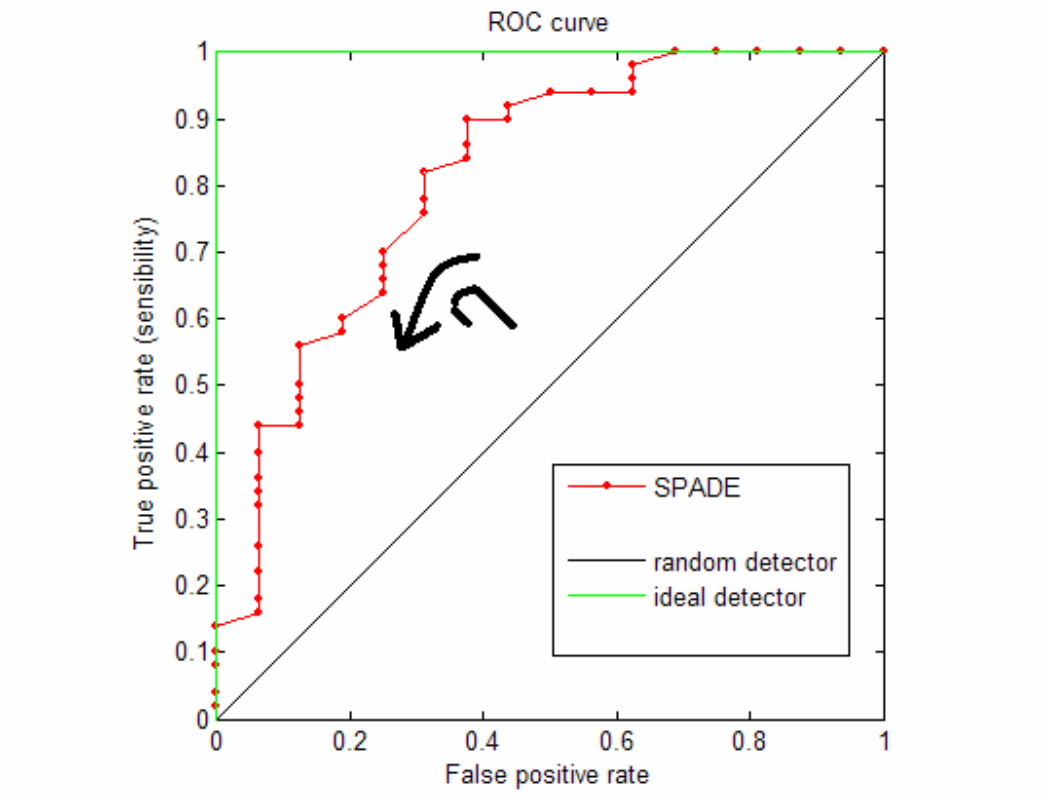
5. Uso de una base de datos.

Uno de los requerimientos más importantes para un IDS es un efectividad, es decir, que detecte la mayoría de intrusiones posibles, manteniendo el porcentaje de falsos positivos en un nivel aceptable. Con los experimentos anteriores no era posible comprobar esto así que para ello se ha utilizado unos datos conocidos.

Se ha utilizado la base de datos 1999 DARPA/MIT Lincoln Laboratory para la detección de intrusiones off-line. Se utilizaron los datos correspondientes a las semanas 4 y 5 donde había 201 instancias de 58 ataques diferentes.

Utilizando esta base de datos, SPADE obtuvo 5162 alarmas por lo que la tasa de falsos positivos es muy alta. Esto se debe a que, en la base de datos, no hay un largo periodo de tráfico "normal" por lo que SPADE no puede construir su base de datos apropiadamente. La solución sería entonces entrenar primero SPADE con un largo periodo de tráfico sin ataques y, así, SPADE podría construir la tabla de probabilidades correctamente.

Con esta misma base de datos se comprobó cómo variaba la eficacia de SPADE con el umbral. Para ello, se utilizó la curva ROC (Receiver Operating Characteristic) que permite comprobar la relación entre las detecciones correctas y los falsos positivos cuando se cambia un parámetro. El umbral se cambió manualmente de 1 a 15 y se obtuvo la siguiente curva:



Se puede ver que a medida que el umbral disminuye, se detectan más ataques pero aumenta el número de falsos positivos. Esto es lógico ya que cuanto más pequeño sea el umbral, más tráfico lo sobrepasará pero será tráfico libre de ataques. Como en todo sistema, existe un compromiso entre la sensibilidad del detector y la tasa de falsos positivos. En esto se basará la calidad del detector. Como se ve en el dibujo, el detector ideal sería el que tuviese su curva ROC situada donde la verde. Como esto no es posible, se toma un valor para el cual la tasa de detección es buena comparada con la de falsos positivos. En este caso, está situado entorno a un umbral de 7.

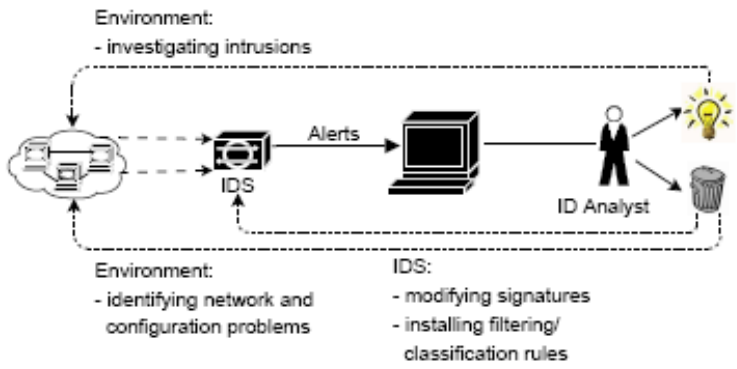
c. Conclusiones

El objetivo de este proyecto era proponer un nuevo IDS para la red de la universidad de Stuttgart ya que el utilizan actualmente, llamado Peakflow, es un IDS basado en usos indebidos y no es capaz de detectar ataques no definidos con anterioridad.

Se propusieron dos IDS basados en detección de anomalías: SOM y SPADE. Finalmente se desechó para el estudio el SOM por la necesidad de tener que diseñar el capturador del tráfico y la necesidad de tener un gran y significativo conjunto de entrenamiento. Nos decantamos, por lo tanto, por el SPADE.

SPADE detecta ataques basándose en probabilidades y decide a través de un umbral que se adapta con los últimos resultados. SPADE resuelve el problema de los ataques noveles pero se ha visto que no es siempre eficiente porque considera tráfico anormal como normal cuando los ataques son continuos o cuando no hay suficiente tráfico normal para calcular las probabilidades correctamente e introduce una tasa alta de falsas alarmas.

Se ha comprobado por lo tanto que ni los algoritmos basados en anomalías ni los basados en usos indebidos son totalmente eficientes para la detección de ataques en la red. Se puede entonces optar por un sistema mixto utilizando los dos tipos de detectores. Sin embargo, parece que, por el momento, la actuación del humano para una decisión final, es necesaria:



X. References

[1] Symantec Global Internet Security Threat Report, Symantec, Trends for July–December 07, Volume XII, Published April 2008.

[2] Lan K., Hussain A. and Dutta D., Effect of Malicious Traffic on the Network, Proceedings of Passive and Active Measurement Workshop, *ISI, Marina Del Rey* ,April 2003.

[3] G.Salomon, Protecting against DoS Attacks, 2005, RADWARE.

[4] *Peakflow SP Arbor networks response to joint Committee of the Higher Education and Entertainment Communities Technology Task Force RFI for Technology Opportunities for Addressing Issues Associated with Peer-to-Peer File Sharing on the University and College Campus*, June 2, 2003. http://www.educause.edu/elements/attachments/rfi/rfi_1/Arbor_original.pdf

[5] Joe Stewart , *Witty Worm Analysis*. March 20, 2004, http://www.secureworks.com/research/threats/witty

[6] Mohammed Attik, Laurent Bougrain and Frédéric Alexandre, *Artificial neuronal networks: biological inspirations*, 2005.

[7] Acosta M.I., Salazar J., Zuluaga C., *Tutorial de redes neuronales*, Universidad Tecnológica de Pereira, 2000, http://ohm.utp.edu.co/neuronales/

[8] http://www.sans.org/resources/idfaq/statistic_ids.php

[9] Bradley T., Introduction to Intrusion Detection Systems (IDS), http://netsecurity.about.com/cs/hackertools/a/aa030504.htm

[10] http://es.wikipedia.org/wiki/Red_neuronal_artificial

[11] Nancy A. Durgin and Pengchu Zhang, Sandia National Laboratorios, Profile-Based Adaptive Anomaly Detection for Network Security, November 2005

[12] Zanero S., Analysing TCP Traffic Patterns Using Self Organizing Maps, D.E.I.-Politecnico di Milano.

[13] R. Lippmann, et al., "The 1999 DARPA Off-Line Intrusion Detection Evaluation", Computer Networks 34(4) 579-595, 2000. Data is available at http://www.ll.mit.edu/IST/ideval/