Title:	Document Version:
Technical Report TR4.1A.9 **IPv6 Network Management and Operation Tools**	0.5

Project Number:	Project Acronym:	Project Title:
IST-2001-32161	Euro6IX	European IPv6 Internet Exchanges Backbone

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type* - Security**:
31/12/2003	25/02/2003	R – PU

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other
** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Responsible and Editor/Author:	Organization:	Contributing WP:
Aurora Ferrándiz	TID	WP4

Authors (organizations) in alphabetical order:

Eduardo Azañón (TID), Ignacio Grande (TID), Francisco Romero (TID), Carlos Ralli (TID).

Abstract:

This document explains the design and operation of TID's applications for IPv6 Network Management and Security Control tools: Magalia and Topaz. These sub-activities are inside the context of A4.2 in WP4.

Keywords:

Attacks against IPv6 networks, Detection Logic, Lesstif, Magalia, Management Console, module_hostlive, module_SNMP, NIDS, Network Management and Operation, Patterns, RADIUS, Rules, Scotty, Sensors, Topaz, Xges.

© Euro6IX Consortium

Revision History

Revision	Date	Description	Author (Organization)
v0.1	15/12/2002	First version	Aurora Ferrándiz (TID)
v0.2	10/01/2003	Document updated to match project template	David Fernández (UPM)
v0.3	14/01/2003	Final corrections	Aurora Ferrándiz (TID)
v0.4	18/02/2003	Final revision	David Fernández (UPM)
v0.5	25/02/2003	Logos added an PDF generated	Jordi Palet (Consulintel)

Executive Summary

This document describes the actual status of the main activities of software implementations at TID labs: Magalia and Topaz.

The development of these two applications provide a new point of view in the implementation of these kind of systems and try to adapt the software development to the real necessities of IPv6 networks.

Table of Contents

1. Introduction ***6***

2. MAGALIA ***8***

2.1 Introduction **8**

2.2 Magalia architecture and interaction with other modules. **10**

2.3 Description of the general operation of the application and interactions among modules. **12**

3. TOPAZ ***16***

3.1.1 System specification 18

3.2 Topaz software architecture design **19**

3.2.1 Sensor 19

3.2.2 Management Console 21

3.2.2.1 Management Console Kernel 21

3.2.2.2 Management Console Graphical Interface 22

3.2.3 Protocol of Alerts 23

3.2.4 Detection Logic 24

3.2.5 Format of Detection Logic in Console Management 24

3.2.6 Format of Detection Logic in a sensor 24

4. Summary and Conclusions ***25***

4.1 MAGALIA **25**

4.2 TOPAZ **25**

5. Future Work ***26***

5.1 MAGALIA **26**

5.2 TOPAZ **26**

6. References ***28***

Table of Figures

Figure 2-1: Start of the application. The whole IPv6 network ***8***
Figure 2-2: MAD6IX ***9***
Figure 2-3: MAD6IX Detailed Context Map ***9***
Figure 2-4: Magalia's basic architecture. ***10***
Figure 2-5: Magalia operation. ***14***
Figure 3-1: Overview of an IDS general operation ***17***
Figure 3-2: Topaz software architecture design ***19***
Figure 3-3: Sensor software architecture ***20***
Figure 3-4: Management Console Software architecture ***22***
Figure 3-5: Events received by the Graphical Interface ***23***

1. INTRODUCTION

During first year two management applications have been developed at TID labs: Magalia and Topaz.

Magalia is designed to have an SNMP monitor and management system of IPv6 networks. This application has been developed starting from zero (not adapting a free source one) to allow the implementation of an innovative modular design with new features to test within Euro6IX.

More exactly, Magalia will implement a distributed model, i.e. each IX runs a Magalia instance that sends authorized information to others. This feature is required in Euro6IX since the backbone is made of networks owned and managed by different Telcos but the experience will work also in other networks where "shared management" is required.

All of this control/monitoring over the network is made using IPv6 connections and SNMP queries. Thanks to Magalia modular design, the modules that perform SNMP queries may use SNMP IPv4 until a concrete equipment supports native v6 queries. The modules transmit the results to the application kernel using IPv6, so it is possible to manage IPv4 equipments remotely through an IPv6 network.

New features developed during 2Y1:

- Context Maps in Graphical Interface,
- Communication between Graphical Interface-Magalia Kernel,
- Communication between External Modules-Magalia Kernel d
- External Modules implemented:
    - Hostlive
    - Module_SNMP

New features planned for 1Y2:

- "Shared Network Management"
- Authentication and/or Encrypted Transactions

Topaz is a Network Intrusion Detection System (NIDS from here on). This application involves analysis, test and diagnostic activities of an IPv6 network, like is specified in WP4.2 from Euro6IX project.

An adaptation of an existing IDS has been evaluated but it has been decided to develop the system starting from zero due to the following reasons:

- A modular specific design is preferred to match Euro6IX requirements.
- The sniffer and basic functionalities software is not a great development since the largest effort in an IDS is spent when "learning" and configure detection patterns (the IDS Logic) and this has to be re-write anyway because IPv6 and IPv4 attack patterns are different).

Work done during 1Y2:

- Network Monitor: Capture and analysis of IPv6 packets
- Implementation of ICMP attacks and connections limit to a port.
- Client Interface development.
- Integration of Network monitor and Client interface.

Next features planned for 2Y1:

- Server failed mechanism. If Management Console falls, the Sensors must know it and wait for the restart of Management Console.
- Detection of new attacks (TID accepts suggestions from the partners).
- Reactions against detected attacks.
- Messages between Sensors and Management Console in an encrypted way.
- Remote management of Sensors.

2. MAGALIA

2.1 Introduction

Magalia is TID's offer for the management, operation and maintenance of IPv6 networks of Euro6IX project. It's an application developed in Linux and Solaris 8 environments, which provides information about Network State, and allows remote control over each element of the net.

This application has a Lesstif-based **[10]** graphical interface that eases its use and provides a general vision of the network composition and the state of each element. The interface is implemented as an independent module called Xges that exchanges signals with the kernel of the application (from here on: Magalia). These signals are from Xges point of view:

- Output signals (from Xges to Magalia): These signals are originated within user-Xges interaction.
- Input signals (from Magalia to Xges): To make a graphical update.

Below there are some snapshots of how the graphical interface looks like:

The whole Euro6IX backbone appears when starting the graphical interface drawn over the map of Europe (**Skulls icons mean no connectivity from Magalia's kernel**).

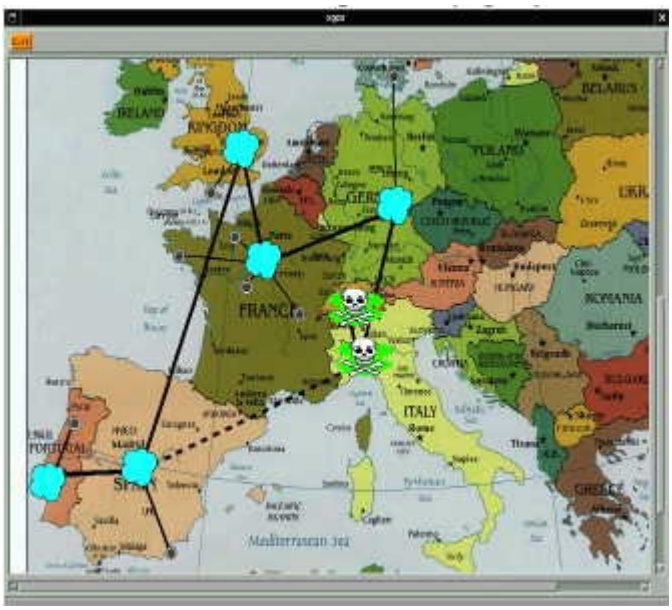

Figure 2-1: Start of the application. The whole IPv6 network

Clicking over MAD6IX with the right button, a new network context is opened. In this example, the Spanish IX:

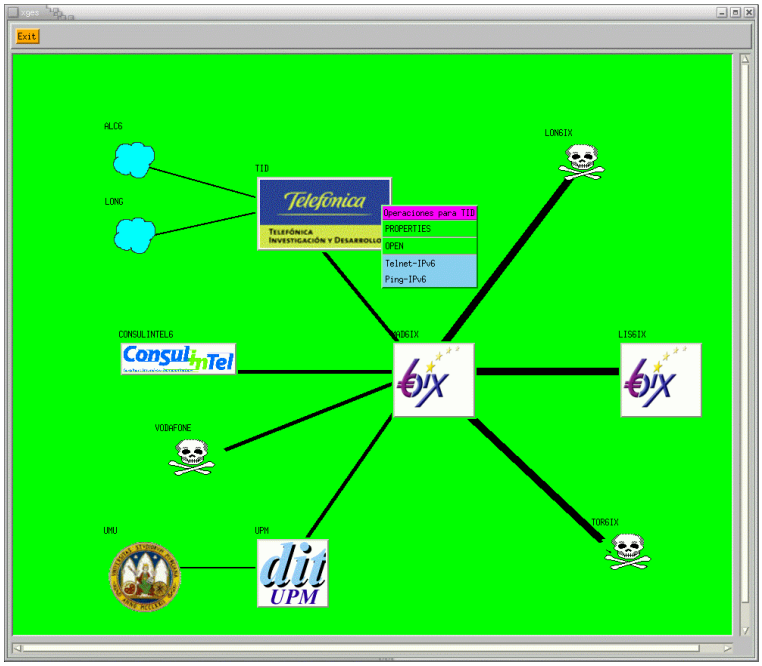

Figure 2-2: MAD6IX

Again, if clicking right button over any of the elements with connectivity, more details of the selected item are showed, and so on until the last level of detail (in next releases, depending on user's privileges to access to the different nets).

This way, a user of the IPv6 TID network can see his own network but not the others.

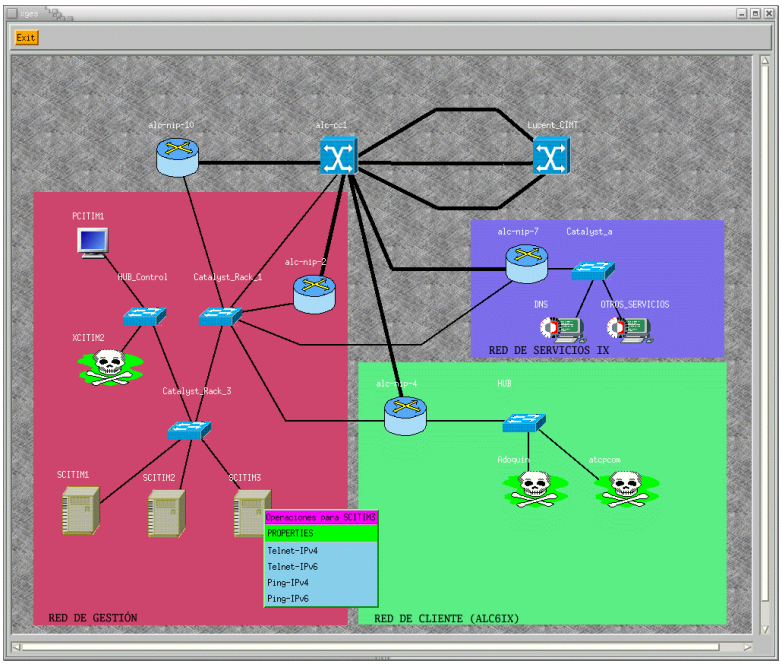

Figure 2-3: MAD6IX Detailed Context Map

2.2 Magalia architecture and interaction with other modules.

Magalia's basic architecture is represented in the next scheme:

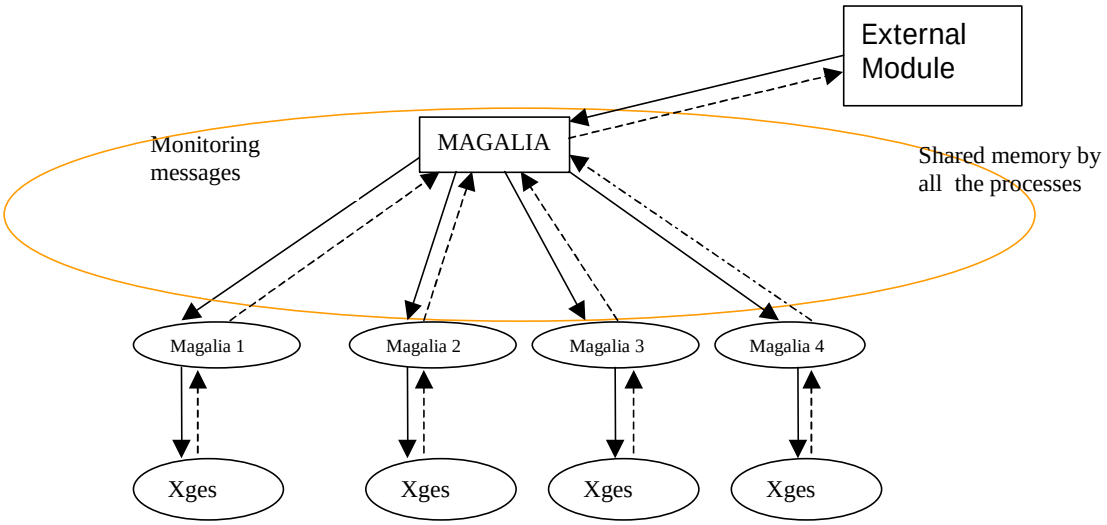

Figure 2-4: Magalia's basic architecture.

A brief description of each module follows:

- **Xges**, graphical interface: Reads and interprets the FTR (it is the acronym for Network Topology File, in Spanish) and represents it graphically. It also manages the interaction between the application and the network operator.

- **Magalia**, kernel of the application: Receives notifications of external modules, process them (modifying the FTR if that's the point) and notifies Xges if there have been modifications in the FTR.

  In turn, Magalia receives Xges' notifications due to the interaction between the user and the graphs and menus of the display.

  Magalia is a set of processes executed as daemons in one or more network administration machines.

  Magalia has a proper configuration file that contains specific values needed for the correct execution of the processes.

  In next releases some kind of communication facilities between different Magalia kernels will be implemented to allow the monitoring of shared networks.

- **Plugins**: They are libraries of functions linked dynamically to Xges.

  They modify the behaviour of the graphical interface.

  Basically they are translators of Magalia's messages to Xges' low-level messages, and functions that extend Xges' API.

- **External Modules**: They accomplish a very concrete and well-defined operation over certain elements of the net, generally by means of SNMP consultations **[3] [4].**

  Modules are configured in a similar way to Xges configuration, i.e., reading an initial configuration file with starting parameters. These parameters help the module to find where the server is running.

  Once the module is started, it's identified by the kernel, the kernel provides, if is allowed to, the networks/elements to watch.

  After connection to the kernel, the module asks for the FTR of the network that has to be watched by it. This provokes its enter in an infinite loop watching that network.

  These modules didn't need to be running in the same machine as the kernel does. In fact, its most interesting feature is that some of these modules act as probes in machines of other partners agree with the partner responsible of the probe.

  This feature is very useful to control elements of the IPv6 network not reachable from a local management network.

  The connection with Magalia is monitored in every moment with a polling scheme.

Plugins available:

- libxges_default_t.so : Messages translator from Magalia to Xges.

Examples of external modules (currently available):

- **Hostlive module:** This module watches certain nodes of the network by tracing pings to those nodes **[11].**

  When a node doesn't answer to a ping, this module warms Magalia. Magalia must notify to all Xges instances to change the icon of the dead node into a skull.

  When the node is alive again, it answers to the ping and then warms Magalia again. In this case, Magalia must notify to all Xges instances to replace the skull with the original node icon.

- **Snmp module:** This module monitories the load of certain nodes' links using snmp queries. The module says to Magalia the load (%) of the elected links.

  Magalia translate this information into a colours code for the links and sends this information to Xges instances.

  The kernel of the module is based on the NET-SNMP 5.0.1 library **[6]**, which is able to make IPv4 and IPv6 queries **[1] [2] [3] [4] [5]**.

  The module sends Xges link colour commands directly, without using the translation mechanism.

Configuration of the external modules

The configuration of the external modules is done from the FTR file.

To configure the external modules, these are the necessary steps:

1. The module opens a connection to Magalia
2. Magalia identifies the session against the Radius **[7] [8] [9]**
3. The Radius response is sent to the module
4. The module identifies the networks to be watched.
5. The module asks for one or more FTR's according the Radius information
6. The module searches the nodes that requires surveillance

Each module must have a proper set of variables configured for nodes and links in the FTR.

2.3 Description of the general operation of the application and interactions among modules.

Magalia is a set of processes executed as daemons in one or more network administration machines.

When Magalia begins its execution, Magalia is created as the process father to coordinate the communications between the different instances of Xges and modules. Each one of these instances will be executed in a remote way and connect to the machine where Magalia process is running.

Magalia has a proper configuration file that contains specific values needed for the correct execution of the processes.

The process father is the one that receives the alarms of the external modules or clients and who modifies the FTR. Later it will inform to all of the processes about which modifications in the FTR have been made in order to Xges modify the graphical representation of the network.

The FTR belongs to Magalia's files system. In shared memory only are the modifications of the FTR that has made by the process father and to be sent to Xges

In shared memory there are a list of certain messages that will be sent to the new instances of Xges connected to the kernel to have its information updated since the moment they are started.

Besides, there is a table of processes with the appropriate data for Magalia and its children to be connected and coordinated.

Magalia's children are independent from its father and communicate with Magalia by means of mechanisms of AF_UNIX sockets and shared memory.

1. For a correct operation of the graphical module, Xges will first have to verify if Magalia is really running, since it does not make any sense having Xges working if the kernel of the application is not doing so.

2. Xges will authenticate to access Magalia. For the identification we'll use a RADIUS sever to whom Magalia consults on the user who is trying to operate with Xges.

3. The RADIUS server tells Xges which network topology has to represent. Magalia sets a registry for communications in shared memory for every child process of its can coordinate, and assigns permissions to every child:

    - The administrative user
    - Or the normal user.

    The administrative user can modify parameters of the net and broadcast then; the normal user only can view passively the net.

4. Xges asks for the adequate FTR to Magalia and Magalia sends it via TCP.

5. Xges represents the information of the FTR over a bitmap, Europe's political map by default, though the same module gives the option to the user to choose the layout bitmap or a flat colour background.

    Once initiated the process and authenticated the user who is going to manipulate the tool, the following step is to load the layout image on which the elements of the net are represented.

6. To represent the elements of net, it is necessary to have access to the FTR. Xges will request the kernel of the application (Magalia) this file, that Magalia will send by the channel of communication established between both.

7. Once the access to the FTR is achieved, Xges will interpret each element contained in it and will proceed to its graphical representation. To do that, it uses the graphical libraries for Linux Lestiff and Xpm.

    The generation of the map of the net is carried out by means of interpreting the guidelines of the file FTR (text plain format) and of the syntax of the tool of free distribution Scotty/Tkined **[12]**. Therefore, the initial map is generated with the above mentioned tool.

8. From here on, Xges enters a processing loop of the events; on the one hand the ones provoked by the user interaction which will have to be sent to Magalia

    On the other hand, from Magalia it receives the changes in the state and operation of the different elements and that must be reflected in the representation of the net (partial updates notified to Xges).

9. Magalia is always listening to events from Xges due to the interaction of the user, which could provoke the execution of some external module. Magalia is also listening to messages from the external modules relating to changes in the configuration of the net and that must be processed and sent to Xges

The modification of the information contained in the FTR is accomplished in the following way:

- When an external module detects that there are some elements that have changed its state (the value of one of its associate properties has changed), it sends to Magalia a tupla containing:

[name_of_the_element, attribute_that_has_changed, new_value]
or
[name_of_the_element, attribute_that_has_changed, $value_1$|$value_2$|...|$value_n$]

- Magalia gathers this information stored in a socket of the family AF_UNIX

- "Name_of_the_element" will be the univocal name by which the element is known in the network. To make a correct modification of the graphical representation of this element, Xges needs to know the name used by Scotty/Tkined. Magalia will entrust the accomplishment of this correspondence.

- As soon as Magalia receives the notification, it has to discover the elements of the net affected by the change of the other one. For it, Magalia uses a proper logic file and a configuration file and arranges the opportune changes in the affected element and the related ones.

- Besides, Magalia will also modify the very lines of the FTR concerning to the element and the related ones, writing the new values in, as well as saving it to disc and reading it from memory again.

- The following step executed by Magalia is sending **Update(element)**'s messages to Xges, where **element** is the name used in Scotty/Tkined to refer to name_of_the_element and the related ones.

  Xges will know, this way, that it has to verify the properties associated with "element" That comes as a parameter in every message "update", and realize a partial update over its corresponding argument.

- Finally, Magalia sends to the external module, OK's notification + additional information, if the information has been received correctly, and a mistake code + additional information if there has been some failure. It has been decided that this mistake code should belong to the application.

  The previous description is represented graphically in the following figure:

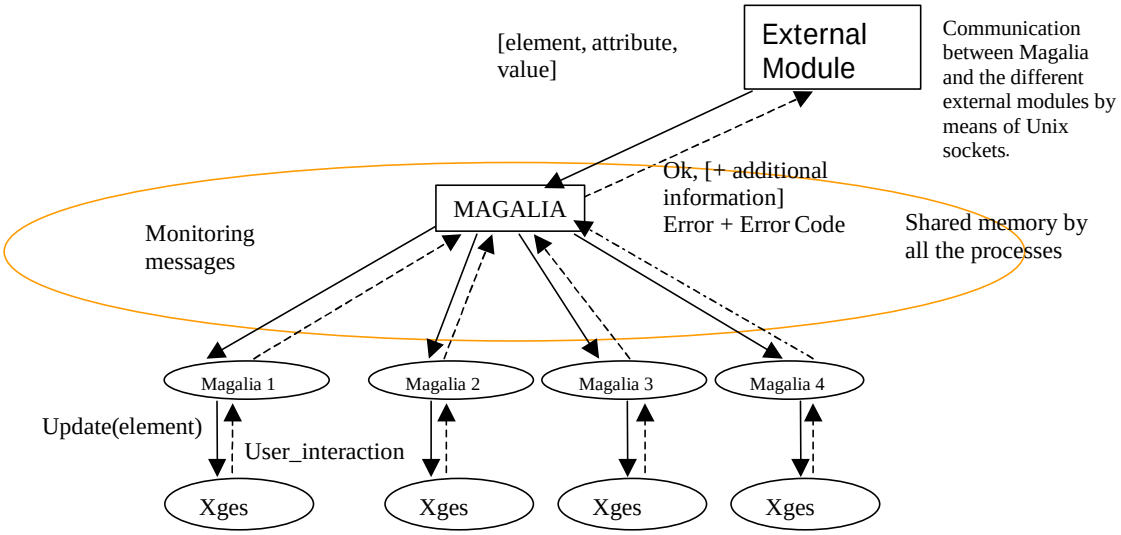

Figure 2-5: Magalia operation.

- If Magalia kernel dies, all the external modules and Xges instances close sockets waiting for the restarting of the Magalia kernel.

  While Magalia is down, Xges instances show a big flashing cross indicating a connection problem.

  After Magalia starts, all the external modules and all the instances of Xges try to connect to the kernel. When success, Xges clears the alarm big cross.

3. TOPAZ

An IDS **[13][14][15]** is a security system that monitors the activity in the nodes and the traffic of a net to obtain certain information that, once analyzed, can be useful in the detection of possible attacks, not only internal, but also external to the organization.

An IDS is the alarm system that complements the use of firewalls in the access to organizations' systems of information and allows the organization to react against any attack

There are two software components of an IDS system, one or more Sensors (detection function) and a Management Console (as a client, supervision function).

Sensors are applications executed in dedicated machines, sited in strategic points of the network, and in a transparent way for all the users.

They monitor and analyze the current state of the net and/or its hosts. In the analysis of the collected information it has a special importance the rules and patterns of suspected traffic database, which have been recollected along the time and experience of previous attacks.

If a sensor determines that the state of the net adjusts to some of the rules or patterns stored in the mentioned database, it must react immediately starting any of the urgency mechanisms and contact, in a secure way, with the Management Console.

The Management Console will send as soon as possible a complete report of the new attack (any information that adjusts to some pattern is considered an attack).

From the Management Console, a network administrator can consult at any time the reports that the Sensors sent.

The Management Console must confirm if the attacks are true (human check if it is a false alarm) and, in case of being a true attack, start the emergency plan against intruders established by the organization (e.g., reconfigure firewall rules, if it's an external attack, and update the patterns database of the *Sensors* with the information of this new attack).

There are two main kinds of IDS:

- A **Host Intrusion Detection System** (or Host IDS, from here on HIDS) is an intrusion detection system specific for hosts.

  The HDIS has only one Sensor executed in the host itself, and the monitor and analysis of the state consists in the detection of changes in the most important system files and applications, log files, config files, etc.

- On the other hand, **Network Intrusión Detection System** (or Network IDS, from here on NIDS) is a concrete IDS for complete networks.

  Unlike the HIDS, a NIDS can have multiples Sensors sited in strategic places of the net, as the point of entry of the organization firewall, watching the access to critical points of the net.

NDIS monitor and analyze the traffic of the communication net. Because of this it is necessary to make packet capture.

TID's proposal: TOPAZ

Topaz is TID's proposal for implementing a Network Intrusion Detection System (NIDS) over IPv6 nets.

Topaz is a NIDS application that consists in a Sensor and a Management Console to detect intruders in an IPv6 network, using monitoring and IP packet capture analysis.

These are some of the requirements for TOPAZ:

- The IPv6 packet capture analyzed by Topaz's Sensors will come from the Ethernet level.
- The Management Console is a graphical module that allows to the network administrator see a correct analysis of the alerts, suspect packets, and any information useful for the network administrator provided by the Sensor module.
- The intruders' logic consists of simple text files, which contains rules, traces or patterns of previous attacks

Graphically, it can be viewed as follows:

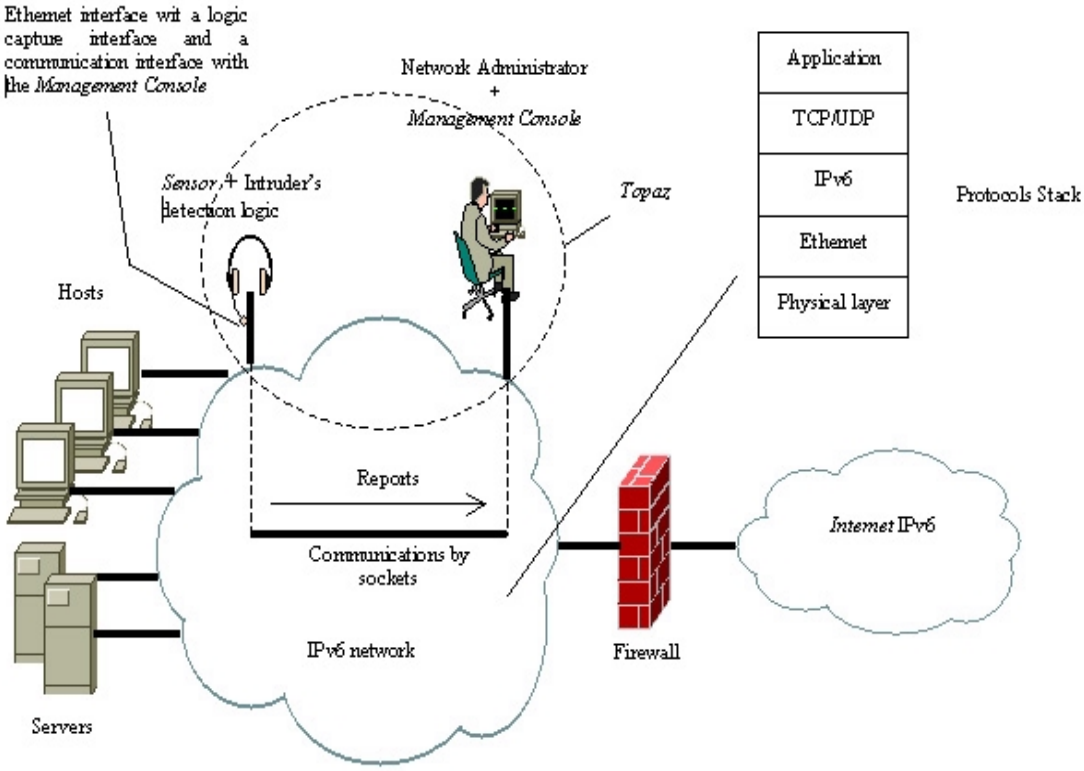

Figure 3-1: Overview of an IDS general operation

3.1.1 System specification

One of the main characteristics of *Topaz* system is the possibility to run over IPv4 and IPv6 networks simultaneously. *Sensors* can communicate with the *Management Console* using both IP protocol versions.

Topaz serves the *Sensors System* using any of its configured logic interfaces, in both IP protocol versions. *Topaz* uses a static UDP port to identify its data units at application layer too.

Topaz's Sensors System can detect, at this moment, two kinds of attacks in an intrusiveness way and taking by reference a set of rules called ***Detection Logic:***

- ICMP attack: if ICMP echo messages are sent to anyone from a configured set of IP layer-restricted machines;
- Limit of connections: if a configured connection limit is exceeded when using specific TCP/UDP ports.

The objective is to send alerts to the *Management Console*, so they can be displayed in a centralised way.

IP datagrams headers analysed by *Topaz*'s sensors are extracted from Ethernet[1] frames. Header fields inspected are *Protocol* field (IPv4), *Next Header* field (IPv6) and TCP/UDP destination port field in TCP/UDP segments (these are contained in IP datagrams).

Sensors have been developed using C++ **[17]** programming language, in both *LINUX* and *Windows* platforms. This assures platform independence and application portability.

The *Management Console* is a traditional *Windows* event-oriented application, adding graphical capabilities and alert analysis, and tracing functionalities **[18]** (in general, all the useful information provided by *Sensors System*).

Network-layer communication between *Management Console* and the *Sensors System* is an important and critical question.

Usually, a VLAN network is used to isolate both types server and clients. However, the use of VLAN networks implies the use of distribution of topology and membership registration protocols in which we are not interested, so they are not considered.

In this way, the need of security has been moved from network layer to application layer by defining an own application protocol, called *Protocol of Alerts*. This protocol considers all messages exchanges combinations between a *sensor* and the *Management Console*, in terms of *sensor* connection and management, logic distributions and updates, alerts reports, etc.

Topaz generates two files with useful information for the *Network Administrator*. The first one is an ASCII file containing captured frames from the network, and the *sensors* provide it. The second one is provided by both *Management Console* and *Sensors System*, and consists in traces about the *Protocol of Alerts*, like a log file.

[1] Ethernet networks are mandatory in *Topaz* infrastructure because they are the most frequent and easiest corporative networks. Another network sublayers and protocols, like ATM or FR, or MPLS encapsulations will be considered in next releases.

Another characteristic that makes *Topaz* NIDS a bit different from common NIDS is its individualised management of *Detection Logic*. Each sensor of *Sensors System* can be independently configured, what gives high functionality and flexibility. This set of *Detection Logic* is maintained by *Management Console* and they are distributed to sensors when they connect at first time.

If *Detection Logic* is modified, the new (or useless) rules are sent only to those connected sensors that must use (or used) these rules.

In this way, updates do not decrease the NIDS efficiency.

3.2 Topaz software architecture design

The components of Topaz are two:

- The ***Sensors***, that act as clients of
- The **Management Console**, which is *Sensors'* server. It has a kernel and an interface.

The way these components interact is represented in **Figure 3-2.**

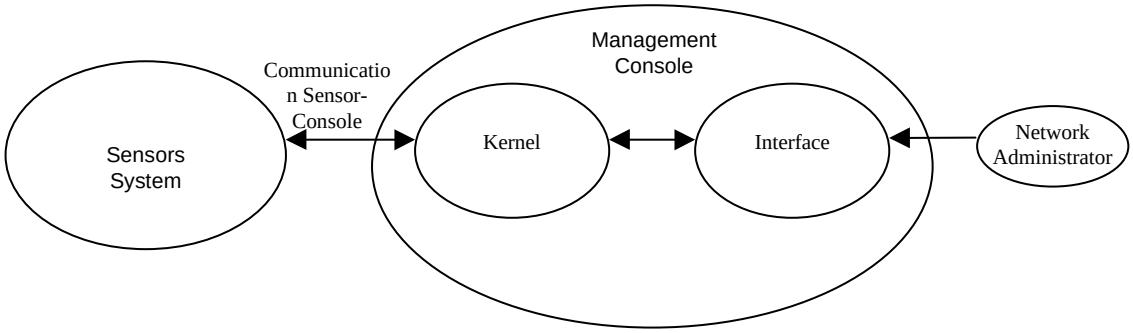

Figure 3-2: Topaz software architecture design

The communication between *Sensors* and the *Management Console* is established by using a protocol defined between them, the *Protocol of Alerts* via UDP sockets **[18] [19] [20]**.

3.2.1 Sensor

Its function is the capture of Ethernet frames to get an analysis of the IP packets contained in every Ethernet frame, and to get the messages that come from the *Management Console* using the *Protocol of Alerts*.

This is why there are two different modules, one for Ethernet frames capture and analysis, the ***Detector***, and the other one dedicated to manage the communications with the *Management Console*, the ***Messages Manager*** (in **Figure 3-3** it is showed the way both modules interact).

- The *Detector* uses an interface based on the library wpcap.lib **[16]** for C/C++ in Windows and pcap.lib for Linux.

  The *Detector* module uses this interface to make a copy in local memory of the Ethernet frames looking for:

- The ICMP identifier of an IP datagram trying to reach an ICMP-inaccessible host,
- Or the number of a TCP/UDP port with limited number of connections allowed.

When an attack is detected, the module must send an **ALERT**-type message to notify immediately about it to the *Management Console*.

The protocol used to communicate *Sensors* and *Management Console* is called *Protocol of Alerts.*

- There is another module called *Messages Manager*. It is the responsible to connect a *sensor* to the *Management Console*. It also answers to the polling request made by the Console, and attends to the first distribution and updates of *Detection Logic*.

A *sensor* is identified by three parameters:

1. A name chosen by the Network Administrator,
2. IP address of the host where is running,
3. The number of the UDP local port which it is connected to the *Management Console*.

To establish a successful connection, the s*ensor* must be registered in the *Management Console* with this information correctly configured. Otherwise, the *Network Administrator* is the responsible to allow a new connection from an unregistered *sensor*.

A *sensor* always provides as output two files: the frames captured in ASCII format and a log file with traces of messages sent to the *Management Console*.

The *sensors* don't take any action against any attack. Its only mission is to notify the *Management Console* about them.

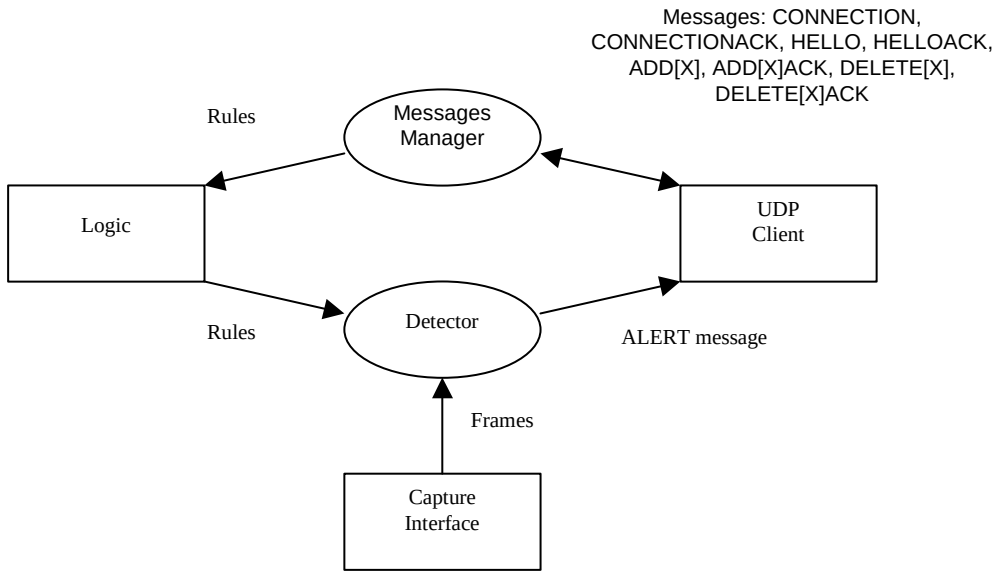

Figure 3-3: Sensor software architecture

3.2.2 Management Console

The *Management Console* allows to the Network Administrator the possibility to see the reports of the *Sensors* and at the same time configure every *sensor* in a graphical way Windows-styled.

3.2.2.1 Management Console Kernel

The *Management Console* receives and displays the messages sent by the *sensors*.

In the *Management Console* side, a concurrent treatment is needed due to there are more functionalities that requires simultaneous treatment, as:

- The management of *sensors* connection, that is, the admission and register of a non-registered *sensor* and the connection of registered *sensors*.
- The processing of received alerts.
- The distribution of the *Detection Logic* updates.

These are the modules of the *Management Console:*

- ***Alerts Manager:*** This module receives the alerts of the attacks detected by the *Sensors*. It processes and displays them to the *Network Administrator*.
- The **Agent of Connections** receives and attends to connections demands made by the *sensors*, checking if a new *sensor* trying to connect is already registered, or otherwise registering it.

  Besides, to test the state of the connected *sensors*, a special kind of message it is implemented. This message is sent to the connected *sensors* periodically, and if a *sensor* doesn't answer it means that it is not live yet.
- The **Logic Manager** makes the distribution of the *Detection Logic* and its updates**.**

  When a *sensor* connects the first time, the distribution can be done in two different ways:

  1. If the *sensor* is unregistered in the *Management Console*, a default copy of the *Detection Logic* (a general version) is sent to the new *sensor*.
  2. If the *sensor* is already registered, only a particularised version is sent.

  The distributions of updates are of the second type.
- The ***Messages Centre*** takes all of the messages that come from the *sensors* and classifies them into the correct buffer for every module. Thanks to this module it is possible to implement the concurrency required for the *Management Console.*

The last responsible of any decision to take against any attack, is the *Network Administrator*.

The communication between all these modules is represented graphically in **Figure 3-4**.

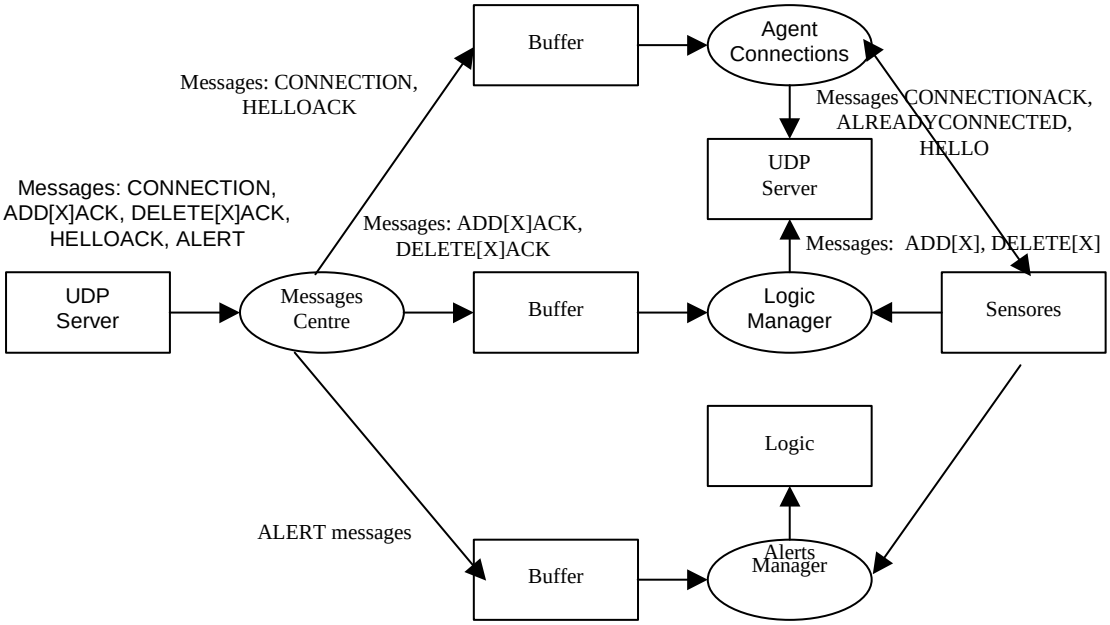

Figure 3-4: Management Console Software architecture

3.2.2.2 Management Console Graphical Interface

The graphical interface is event-based.

External events (the user changes *sensor* configuration options) to Topaz's interface are:

- Add/delete a registered *sensor*.
- Add/delete a category of rules.
- Add/delete a rule.

In the other hand we have internal events, i.e., events communicated by the *sensors* to the kernel of the *Management Console* that have to be showed on screen.

Some internal events are:

- Attack notifications.
- Updates of the list of *sensors* connected to the *Management Console*.
- Non–allowed connections.

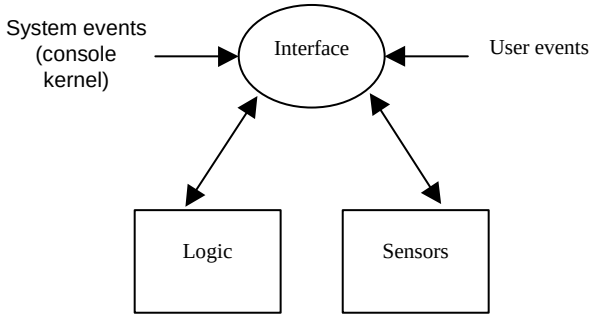

Figure 3-5: Events received by the Graphical Interface

3.2.3 Protocol of Alerts

The protocol developed for distributed communication between sensors and Management Console is based in the interchange of messages, in a certain order.

Management Console knows which sensors are connected by the following mechanism:

1. The *sensor* sends a CONNECTION-type message to *Management Console*.

2. If connection is possible (there is not a sensor connected with the same identifier yet), *Management Console* sends a CONNECTIONACK-type message to the *sensor*.

It's necessary that *Protocol of Alerts* has some error control mechanism, because the transport protocol used, UDP, does not provide it.

In this way, if the *sensor* does not receive a CONNECTIONACK-type message during a configured time period, it will send again a CONNECTION-type message, either it has never arrived to *Management Console*, or acknowledge has never arrived to the *sensor*.

In addition, it's necessary to add a sequence number in order to know the number of requests, so *Management Console* could know if it's a real double connection or a CONNECTIONACK-type message lost.

If the sensor is already connected, an ALREADYCONNECTED-type message is sent.

Once a sensor is connected, it's remarked so the *Logic Manager* knows it must send to the *sensor* its particularised *Detection Logic* (if it has). This distribution requires:

1. The *Logic Manager* makes an ADD[X]-type message (X refers to actual category of rules) taking by reference the particularised *Detection Logic* of the sensor, if it has, or the default global *Detection Logic*.

2. The sensor receives the ADD[X]-type message and extracts its contained rules. If the information extracted has no errors, it sends an ADD[X]ACK-type message to *Management Console* and updates its local *Detection Logic*.

Like previously described in the first interchange, *Management Console* waits for an ADD[X]ACK-type message during a configured time period, and if it does not arrive in time, an ADD[X]-type message is sent again. It also must have the number of requests.

After this, *Management Console* must wait the arrival of ALERT-type messages. These messages contain a number identifying the attack, the IP directions of the intruder and the attacked host. These messages are never acknowledged, sensors need completely dedication to detection task and can not wait for acknowledges.

Once a sensor has received its *Detection Logic* it's ready to receive updates too. Updates are considered like the initial distribution, i.e., using ADD[X] and ADD[X]ACK-type messages, plus DELETE[X] and DELETE[X]ACK-type messages.

Sensors are polled by *Management Console* too, to confirm their state (reachable/unreachable). This mechanism is easy, *Management Console* sends periodically HELLO-type messages to all connected sensors, and these must answer during a configured time period. In this case, expiration of time period means that the sensor is unreachable, and sequence numbers aren't necessaries because HELLO-type messages must be always acknowledged.

3.2.4 Detection Logic

Detection Logic contains critical information. At this moment, it contains ICMP-inaccessible machines list and connection limit of TCP/UDP ports, i.e. the set of detection rules.

Management Console and *Sensors System* consider *Detection Logic* in a different way. As previously said, *Management Console* maintains a permanent version of every individualised sensor's Detection Logic (if it has, otherwise, it exists a default global Detection Logic). When a sensor connects, it receives a distributed copy of its own logic.

In *Management Console* side, *Detection Logic* is saved as one plain text file per *sensor*. These files contain the set of particularised rules for each sensor.

Sensors maintain *Detection Logic* in local memory, as temporal objects created when they are distributed.

Equivalently, every modification of *Detection Logic* is distributed in the same way.

3.2.5 Format of Detection Logic in Console Management

Management Console manages plain text files to save *Detection Logic*. The internal format of these files is simple, and it is based in the use of key words. So, the possibilities are:

- "ICMP" (means that we can find an ICMP-inaccessible host) + [YES/NO] (means that the machine is accessible/inaccessible by ICMP packets) + IP direction/name of the machine.

- "PORT" (means that we can find a connection limit of TCP/UDP port) + [TCP/UDP] (kind of port) + limit.

3.2.6 Format of Detection Logic in a sensor

Detection Logic's format in *sensors* depends on the design of the C++ class, but, in general, it is a linked chain of objects, each of one, containing a rule or attack definition.

4. SUMMARY AND CONCLUSIONS

4.1 MAGALIA

Development of a beta version of Magalia, with all the functionality described in this document.

This beta version is being tested by UPM, but in a future will be installed by other partners (mainly the Telcos) to monitor their own IPv6 networks.

4.2 TOPAZ

Work done during 1Y2:

- Network Monitor: Capture and analysis of IPv6 packets
- Implementation of ICMP attacks and connections limit to a port.
- Client Interface development.
- Integration of Network monitor and Client interface.

5. FUTURE WORK

5.1 MAGALIA

Next features to be implemented in Magalia:

- "Shared Network Management": In Euro6IX there are different management domains since those networks are owned by different entities. There are two possibilities to manage such a Backbone/network:

    1. Establish a global Management center that accesses all network elements in all networks.

    2. Run a management software instance by each organization.

    Since the Euro6IX looks for real and commercial-like solutions and there are different Telcos and organizations, solution (1) seems difficult to succeed.

    Solution (2) is the one proposed in Euro6IX (running one instance of Magalia in each organization). The main disadvantage is the lack of a global view, since Magalia instance running at TID should not be allowed to get information from FT routers (for instance).

    To solve this, Magalia will try to implement a mechanism to allow different instances of the kernel to interact and share information (configured by network administrators).

    This key feature will imply to add user profiles and it will bring a new research concept in the deployment of Management applications. This feature can be implemented since Magalia has been written from the scratch foreseeing this in its design and architecture.

- Authentication and/or Encryption transactions: Establish authenticated and/or secure communications between Xges and Magalia, as well as different Magalia kernels when performing "Shared Network Management".

This activity will continue next year.

5.2 TOPAZ

Next features planned for 2Y1:

- Server failed mechanism. If Management Console falls, the Sensors must know it and wait for the restart of Management Console. In 2Y1, TID developers will implement this mechanism.

- Detection of new attacks (TID accepts suggestions from the partners). More attacks will be implemented to assure the total security of the network.

- Reactions against detected attacks. Today, no action is taken when an attack is detected. The Management Console will be able to send orders to any attacked host in order to protect it, for example close ports with an exceeded limit of established connections.

- Messages between Sensors and Management Console in an encrypted way. Another functionality to establish secure communication between Sensor and Console is to encrypt/decrypt messages.

- Remote management of Sensors.

This activity will continue next year.

6. REFERENCES

[1] RFC 1155: Structure and Identification of Management Information for TCP/IP-based Internets (http://www.ietf.org/rfc/rfc1155.txt).

[2] RFC 1156: Management Information Base for Network Management of TCP/IP-based internets (http://www.ietf.org/rfc/rfc1156.txt).

[3] RFC 1157: A Simple Network Management Protocol (SNMP) (http://www.ietf.org/rfc/rfc1157.txt).

[4] RFC 1441: Introduction to version 2 of the Internet-standard Network Management Framework

[5] RFC 1442: Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)

[6] NET-SNMP 5.0.1 library: http://net-snmp.sourceforge.net/

[7] RFC 2865: Remote Authentication Dial in User Service (RADIUS). (http://www.ietf.org/rfc/rfc2865.txt)

[8] RFC 2866: RADIUS Accounting. (http://www.ietf.org/rfc/rfc2866.txt)

[9] FreeRadius. Radius Server used in Magalia development and testing. http://www.freeradius.org/

[10] Lesstif. Graphical libraries that Magalia uses. Free clone of OSF/MOTIF libraries (http://www.lesstif.org/). Handbooks for developers: http://www.motifdeveloper.com/.

[11] fping y fping6: Commands used in module_hostlive (http://www.fping.com/)

[12] Scotty/Tkined web: Free source tool to generate network maps.

http://wwwhome.cs.utwente.nl/~schoenw/scotty/

[13] SANS Institute - Computer Security Education and Information Security Training http://www.sans.org/index.php

[14] "Intrusion INC" http://www.intrusion.com/

[15] "NIST Special Publication on Intrusion Detection System" http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf .

[16] Windows Packet Capture Library http://winpcap.polito.it/.

[17] Resources on C++ "Code Project - Free Source Code and Tutorials http://www.codeproject.com

[18] Programming in Windows: http://www.msdn.microsoft.com, http://www.microsoft.com/windows2000/docs/ipv6winsok.doc.

[19] Programming With Berkeley Sockets http://ou800doc.caldera.com/SDK_netapi/CTOC-sockN.intro.html.

[20] RFC 2292 - "Advanced Sockets API for IPv6" (http://www.ietf.org/rfc/rfc2292.txt).