A project and implementation of the testing system for Intrusion Detection Systems. Part II

Andrzej Barczak, Andrzej Orzoł
Institute of Computer Science, University of Podlasie,
St. 3 Maja 54, 08-110 Siedlce, Poland

1 Introduction

Intrusion Detection (ID) is the Security Technology area which tries to identify and isolate intrusions to computer system.

Intrusion Detection is an important component of the system's security, it complies other security technologies. Attacks and intrusions missed by other components or even aimed at them, they can be detected by IDS, by its mechanisms build into, originated in cryptography or artificial intelligence.

In the first part of this article, the introduction to System's Intrusion Detection was described. There were shown sort of attacks, their probable motives and goals. This knowledge helped in specification of choices and evaluation of IDS. The choice of a right product with its right configuration and even its reconfiguration, meaning IDS adjustment to its needs which isn't an easy challenge. A system administrator encounters precise need to test every change in systems configuration of intrusion detection. This process is very demanding and in many cases could become automized by simulation of attacks with directing signature sets or generating of collective statistic in accessible forms. The evaluation of IDS and its configurations requires many tests and that is why at the end of the previous article, the conception of automized system of IDS testing process was shown.

The goal of this article is to show the process of generation of the system which is going to be a tester for intrusion detection. The attention will be focused on the same conception of the IDS tester and solutions for its architecture or used for implementing of a programming language.

During the work on the system, there were many implementation problems which will be described in the right unit.

2 The concept of the tester

The basic condition of full success in choosing the right IDS is stating the needs of safety for a managed system which means leading precise audit of safety. Only a clear and exact rating based on a complex report of audit enables optimal choise of IDS.

Before putting in a new system of intrusion detection, it needs running tests, which as it was shown in the previous article, in the case of the levels of the net are not only difficult to run but also demanding. The testing of different configuration using the same test may take many days. Existing safety scanners give only a draft of possible general security holes in the host, without giving information about important characteristics of IDS resulting from log files analysis.

The testers project, which will in some extend fulfill the need of automatization needed activities leading to increasing efficiency and effectiveness of IDS should include following assumption:

- usage of chosen attacks signature set, leading to simply run simulation of safety evens
- usage of different IDS log files of the net's level
- easy usage of the system by provided simple and intuitive graphic interface
- clear way of collecting data and reports
- possibility of inspect reports by using generally accessible tools such as Internet Browser
- mobility between systems platforms without any main changes in source code
- possible compilations according to chosen platform, being a goal to increasing performance
- the main assumptions of the project leads further to specifying function requirements. Stated functional requirements are shown in the table 2.1. below:

Table 2.1 Functional requirements

Name	Description
Event generator based on rule database	Safety event generator which is using data from accessible rule database as parameters. Row in the database, it is single safety event with correct characteristics.
Usage of existing rules database	Role database should be chosen roles database made of already existing ones. Main characteristic used to choose roles database are: roles database format, methods of licensing, amount of rows and frequency of updating

Usage of IDS based on different log formats	With great amount of multiplicity of IDS, almost each of them uses different log formats for events. The main part of the system cannot change according to tests of different IDS.
The choice of roles for testing	Possibility of choosing the roles, which can be used during running tests. This functionality should be placed in graphic interface.
Information about included rules	Functionality of interface that is reliable on providing information about chosen rule database or its signature
Defining of its own rules	Possibility of building its own set of rules, its own group of rules or modification of existing rules.
Running tests	The basic function of a tester. The test must be run automatically. After its finishing the user must be given the results
Pausing and resuming the test	Leading the test operation enabling pausing of the test for some time and then resuming it.
Stopping of the test	Stopping of the test operation with keeping of all the test results and generating the reports.
Saving the test result	Possibility of saving the test results in the form of two flat files.
Generating the report	Functionality enabling generating of reports with flat files.
Setting up of service addresses	Possibility of configuration of the tester according to service addresses used by HTTP, SMTP, SNMP, AIM and other existing in the local network.
Setting the ports for services	Possibility of defining the ports working of the port that services are working in a given network.
Setting up of the internal network addresses	Possibility of conceding to configuration of the list of addresses, showing actual host in the local network.
Setting up of external network addresses	Functionality enabling configuration of list of addresses in the external network used to running tests.
Intuitive interface	The need for simple intuitive interface for managing the tester.

Non functional requirements resulted from the conception of solving the problem will be shown in the next unit of the article.

3 Considerated solutions of tester's architecture

After setting functional requirements there is a time for thinking about general structure, architecture, that is general conception of system realization. In the case of project teams which is run mostly on brain storm method, based on the knowledge of the discipline. In this part we already know what will be the aim of system's project, so the answer for this question about tasks of the system and his functionality, is clear. During this part arise different conception of system realization. Architecture conception that enables realization of tasks. During the work of tester, four basic results was discussed.

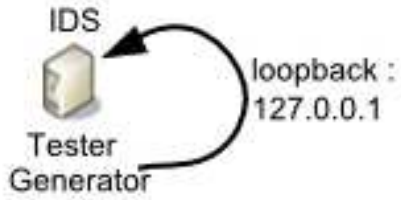


Figure 3.1 Architecture based on single host in the network

The first result shown on the figure 3.1 is based on the simplest of possible conception. On the only one and the same host both IDS and its tester are installed. This approach assumes that IDS will be configurated to lead monitoring on loopback interface (that is 127.0.0.1 addresses). Certainly it leads to many errors when IDS is monitoring manufactural traffic for the address of this virtual interface which leads to generation of many of huge amount of false alarms.

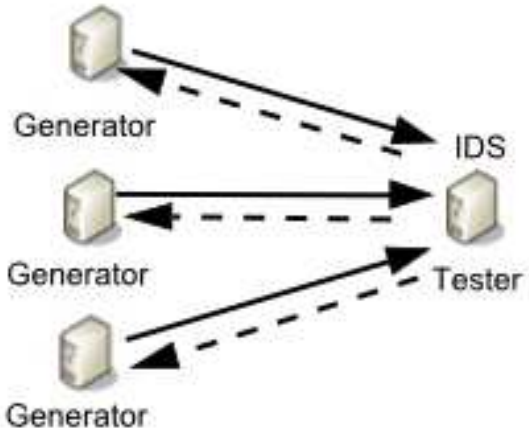


Figure 3.2 Architecture with many generators

In this approach shown on the figure 3.2 on the tester which is placed on the same host in the network as IDS, but this time it manages only one or many generators of events placed in different hosts. This prevents errors known from previous approaches and is almost optimal solution. But after exact recognition

systems specification, on which IDS is installed it shows that they are deprived of graphic interface and one of basic functional requirements was equipping of the tester with friendly graphic interface.

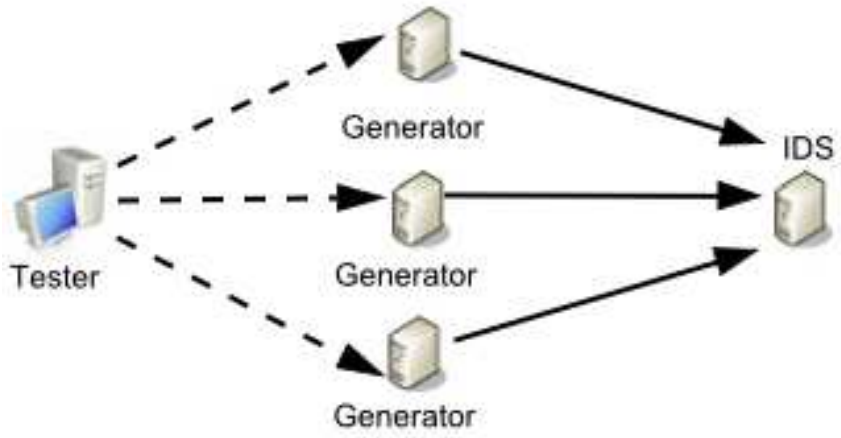


Figure 3.3 Architecture with many generators and a tester on different host with graphic interface

The approach shown in the figure 3.3 eliminates problems related with user graphical interface know from previous approach. On the host with IDS, a sensor is installed, which deals with sending correct data about IDS work to the tester. The tester still manages the network of the generators. This approach has its advantages and disadvantages, it allows for exact simulation of distributed network attack, but from the testers point of view a system forces installation of many generators of host of wide network. The project must be extended by adding a problem solution to organizing manages many equal generators of safety events.

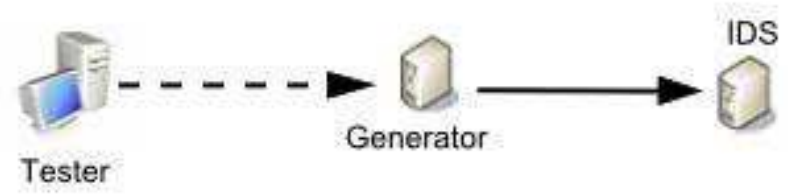


Figure 3.4 Architecture with singular generator on different host with graphic interface

Previous solution after simplifying solution of two problems are shown on figure 3.4. After tests related with falsifying fragments of network IP packet header, shows that many generators can be replaced by one managed by a tester or build into a tester module.

The chosen approach shown on figure 4 confronted with functional requirements gives a quite clear summary of non-functional requirements shown in a table 3.2.

Table 3.2 Non-functional requirements

Name	Description
Usage of rules database SNORT	SNORT is known IDS system based on GPL license. It is difficult to give a number of records in rules database of this system, because of a lot of updating and many non-official sources of rules. It results from simplicity with which SNORT defines its own rules. Actual official free rules database includes around 3000 signatures, but only 2300 is used by tester.
Network architecture	IDS tester cannot be started directly on the same network host as tested IDS because of possibility of interruption of research results. That is way is must be built on architecture client-server. Where a server is a sensor giving the data to a tester.
Compatibility with Java 1.6. language structure	A tester should affirm mobility for its tasks. That is why java is suggested language.
Usage of Swing Library	Swing library gives many graphic components that facilitates building of interface. It also gives compatibility with chosen libraries of JPCap JFreeChart.
Usage of JPCap package	That is extended network service for Java language.
Usage of JFreeChart package	A library with components enabling chart generation important for clear representation of reports data.
Generation of reports to HTML format	HTML documents are serviced by every system that owns internet browser. That is why they were chosen for presentation for test results.
Compatibility of a model with UML 2.0 standard	Usage of UML 2.0 as a standard for models describing IDS tester project.

4 Project and implementation of tester

The major rule in projecting testers system was to division into components. Described structure of division for model system was shown on a diagram of components and interface below (figure 4.1).

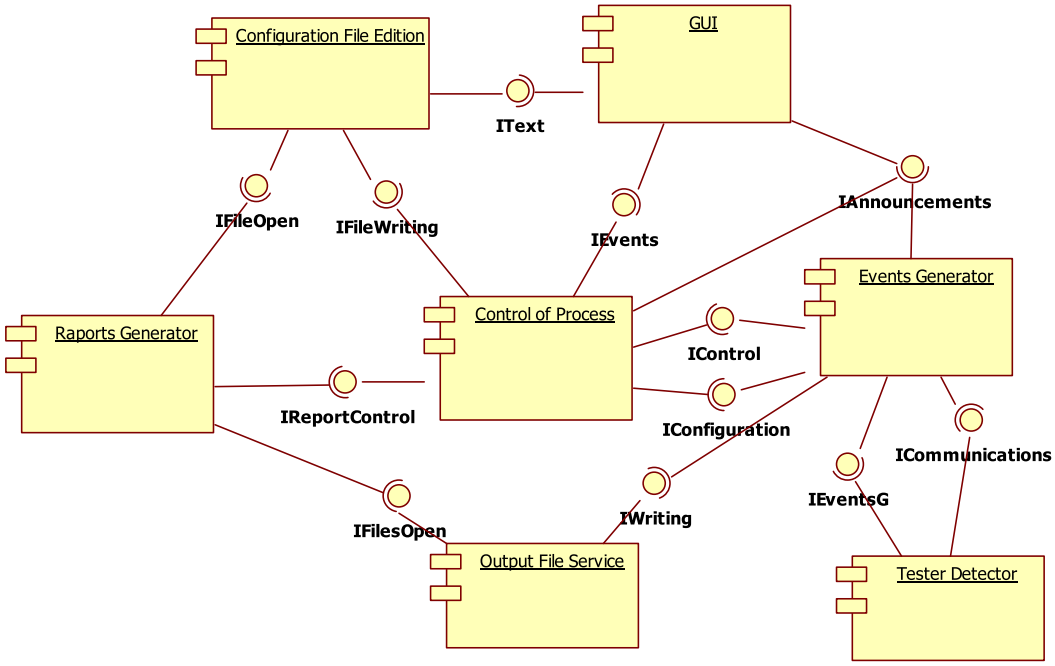


Figure 4.1 A diagram of components for intrusion systems tester

Each of components shown on the diagram is a separated functional model of the system, that is why diagrams of interface were prepared for each one and then class diagrams. An example of interface diagram for GUI component showing graphical users interface below (figure 4.2)

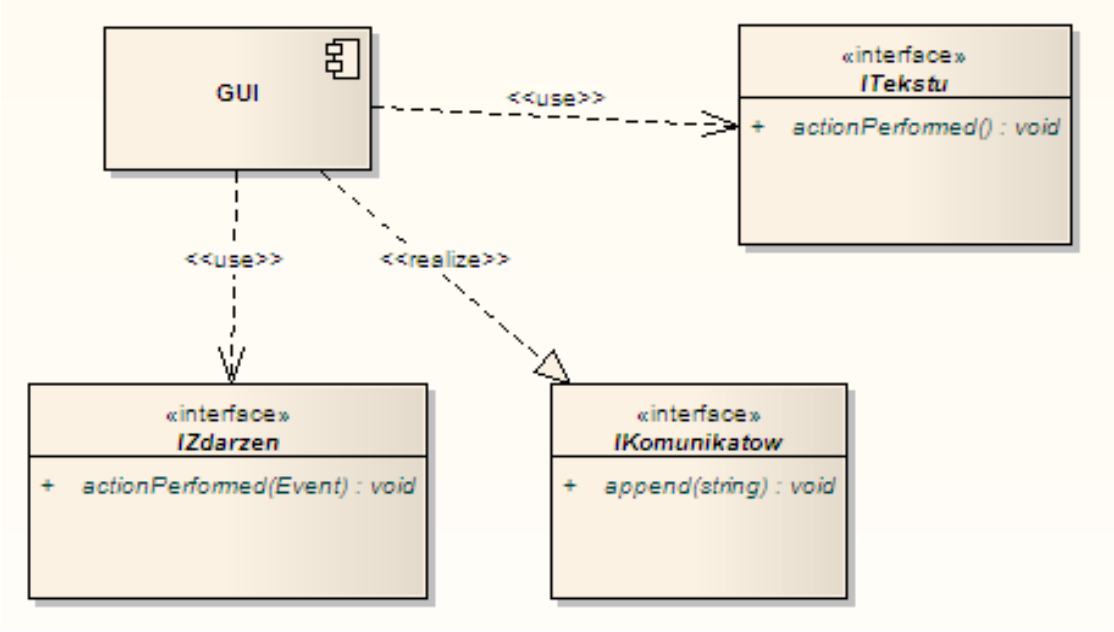


Figure 4.2 The diagram of interface for GUI component

Every model before implementation was described by exact class diagram. The example of class diagram for GUI component was shown on figure 4.3 on each class diagram was situated main class called tester, in order to show relation between functional models and hole system.

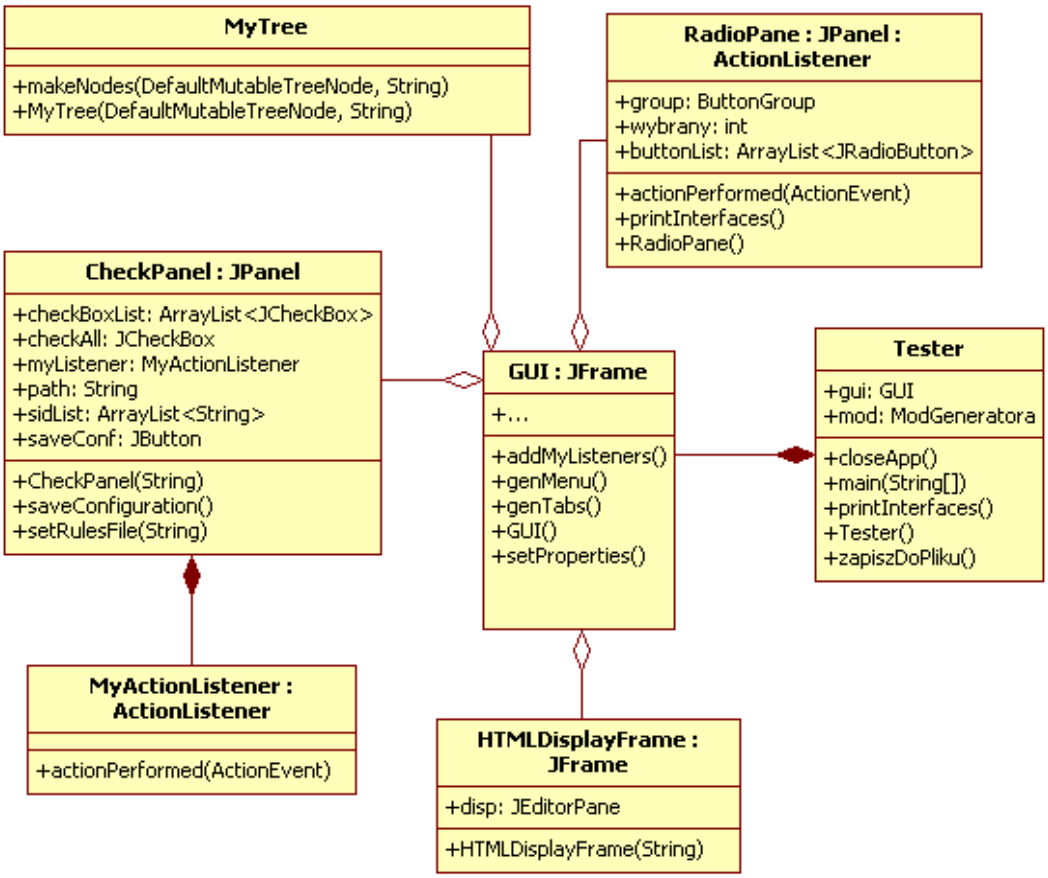


Figure 4.3 Class diagram for GUI component

Before the final implementation, collective class diagram was made for hole system project, which was not put into the article because of its dimensions, as it is a map of hole system. Modeling testers system was based on many different notations from UML language, mainly describing functionality and project dynamic, which will not be shown in this article.

All of described functional system models were implemented correctly, even though there were many problems. The greatest amount of implementation problems accrued when working on model described as event generator, which is the most extended part of testers system. The class called ModPakietu included in this model generating single prepeared network packets has specific build. The build results from the fact that packets, that supposed to be injected by the class to network structure are often prepeared so tools or network services couldn't serve them or serve them incorrectly (ex. Packets for DoS and DDoS types of attacks). The class needed continuous testing during implementation and often its work breached

security rules of virtual Java machine (Java Virtual Machine, JVM) and even operating system kernel.

5 Evaluation of usefulness of tester

The system before first start needs compilation on a chosen host. To run this process properly installation of JPcap, JFreeChart and WinPcap libraries is needed in the case of windows operating systems.

The system was equipped with graphic interface that runs many functions. In the figure 5.1, there is shown tester window during configuration of one of rules data files, enabling the choice of rules, which will be used to run the test.

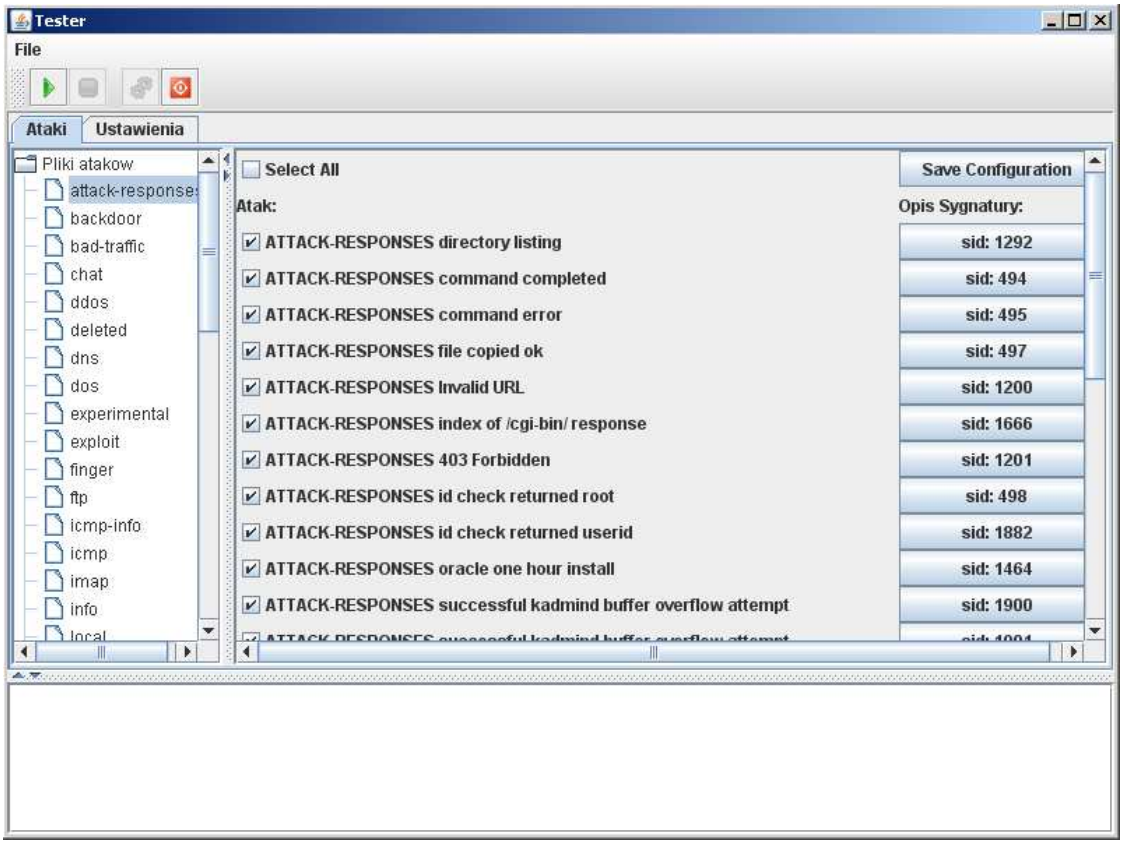
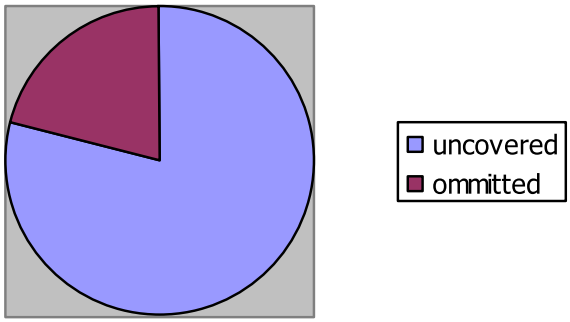


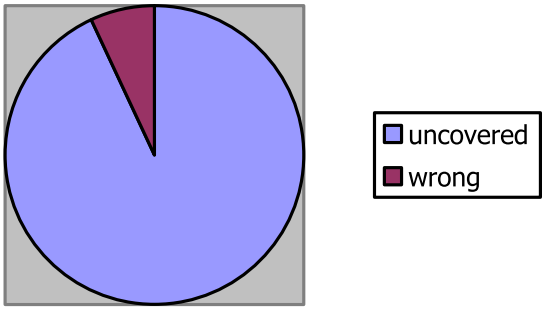
Figure 5.1 Systems tester interface window for intrusion detection: tab “attacks”

Testers interface allows for showing of signature description of chosen attack, which is given with SNORT rules database. Following configuration is available in “properties” tab. The given parameters are configurated:

- IP addresses and ports of chosen network services
- address and port of communication and tester’s sensor
- time delays between generated safety events
- the choice of network interface provided for generator module



a. The efficiency of the uncovered attacks



b. The part of wrong interpreted attacks

Figure 5.2 Summary of run test

The testers interface enables pausing, resuming and stopping of run test in the chosen moment. All reports about run test are written in message console.

To run tests VMware Server 1.0.8. was used, a virtual machine was made with Fedora 10 operating system and SNORT IDS. The test was run in the network with restricted access to external network. SNORT operated on free rules database with excluding "misc.rules" file. Configuration of IP addresses and service ports in the tester was identical with SNORT configuration. The test was running for 25 minutes with delays between events – 500ms. After finishing full test, the system generates reports to HTML file format with charts in PNG format. Those files are compatible with Mozilla Firefox v 3.5 internet browser. The test was run tree times with exact result, shown on figure 5.2.

The report was divided into tree stages. On given figure 5.2 a level of summary is shown. It is general summary of IDS effectivity for all groups of safety events.

Figure 5.3 Snapshoot of opened report: general level

The figure 5.3 shows general level, where only charts of attacks are shown. Detailed level enables browsing of all generated events from chosen group. The data was formatted into table format. Shown data concern:

- event name,
- signature Identification Number (Sid),
- time of generated events,
- delay between time of sending and time of detection,
- the time of event detection,
- detection classification.

6 The chosen tools

Fedora 10 is free operating system from Linux family based on Red Hat architecture. It is the most popular server distribution, on its base for example kernel.org server is working. That is way it was chosen for implementation and running tests.

JFreeChart is free library of java was used for professional presentation of charts. JFreeChart supports many types output data such as Swing components, picture files (PNG and JPEG) and vector graphics (PDF, EPS and SVG). It is open source application distributed on GNU Lesser General Public License (LGPL), which allows for using legally reserved application. Home site of product is www.jfree.org/jfreechart

JDK 1.6 is a software by Sun Microsystems Enterprise, free environment for programming in java language. JDK is characterized by compatibility for many operation systems. The project is on accessible on a GNU GPLv2 licensee.

Jpcap is java library, which enables packets sending through the network. Using Jpcap applications can be projected to capture packets from network, then analyze and visualize using java language. Jpcap is tested on systems: Microsoft Windows (98/2000/XP/Vista), Linux (Fedora, Mandriva, Ubuntu), Mac OS X (Darwin), FreeBSD and Solaris. Jpcap operates on Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP protocols and ICMPv4 packets. It is an open source application on GNU/GPL licensee. Home page of project:
www.jpcap.sourceforge.net

SNORT is Network Intrusion Detection System (NIDS). It gives wide variety of intrusion mechanisms which in real time analyze and register packets based on IP/TCP/UDP/ICMP protocols. Official site of the product is under address www.snort.org

Winpcap is a software enabling sending, capturing and filtering data packets send in local network and the internet. It works on operating platforms Windows NT4/2000/XP/2003/Vista. It was produced by Winpcap Team on GPL licensee. This library is necessary for installation of Jpcap library in Windows environment. Home page of project www.winpacp.org.

7 The conclusions

The project of IDS tester here described was a part of a diploma project titled: “The project and implementation of IDS’s tester”.

During the project realization there were many problems encountered and maid it difficult for satisfying and product. The first decision during realization of tester was the choice of network architecture and optimal rules database for generator. Fallowing the decisions conciderate performance of system project. During realization popular script languages such as Perl decreased its effectiveness. Finally, Java language was chosen, even though having huge packet forms

and libraries, it doesn’t have free interpretor form for regular frases for PCRE language, which is a basis of about 700 phrases in the chosen SNORT rules database.

The problem of continuous testing of results of the work was solved by the help of virtual machine connected in virtual network with free Linux system for Fedora 10 was installed.

The greater amount of work was given to implementation of generator module. Three software languages were used for implementation in order to gain effectiveness, Java language proved to be the most optimal.

The system gained chosen functionality of generator, 2300 safety events generated by SNORT files. At the end only half of the needed independence form the system platform was gained by the system.

Usage of JPcap library, which implementation differs from tested system Windows (windows XP SP3 i386) and Linux (Fedora 10 i386) presses compilation of system on chosen platform.

References

1. "Network Security Tools: Writing, Hacking, and Modifying Security Tools" by Nitesh Dhanjani, Justin Clarke (Paperback - April 4, 2005).
2. "Linux Server Security" by Michael D. Bauer (Paperback - Jan 18, 2005).
3. "Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort" by Michael Rash (Paperback - Sep 15, 2007).
4. "Hacking: The Art of Exploitation, 2nd Edition" by Jon Erickson (Paperback - Jan 11, 2008).
5. "Protect Your Information: With Intrusion Detection" by Alex Lukatsky (Paperback - May 30, 2004).