A High Performance System for Intrusion Detection and Reaction Management

Abdoul Karim Ganame[1], Renaud Bidou[2], Julien Bourgeois[1] and Francois Spies[1]

[1]LIFC, Universite de Franche-Comte
4, place Tharradin, 25211 Montbeliard, France

[2]RADWARE
113 avenue Jean-Baptiste Clement
92100 Boulogne-Billancourt, France

Abstract: Detecting all kinds of intrusions efficiently requires a global view of the monitored network. This can only be achieved with an architecture which is able to gather data from all sources. We have developed a security operation center called SOCBox which is able to detect coordinated attacks that are not detected by traditional IDS. In this article, we present the global architecture of the SOCBox as well as several methods used to test its accuracy and performance. A real ISP network have been used as well as experiments in our lab.

Keywords: IDS, SOC, Distributed intrusion detection, Network security, Graphical cartography center.

1 Introduction

Ensuring network security requires two modules: protection and supervision. Protection is composed of hardware, software and a security policy that must be followed. Even the best protection is always vulnerable to attacks due to unknown security bugs. Besides, the network configuration is subject to constant changes and possibly adds security holes. That is why the network supervision is an essential part of the security process and is realized by security experts.
In order to help the supervisors, Intrusion Detection Systems (IDS) have been developed [3], but these systems have several flaws. First of all, IDSs have an insufficient rate of detection: either too many intrusions are detected or missed [5]. Furthermore, simple IDSs have no sufficient information to detect coordinated attacks. Other types of IDS have been created and tested like distributed one [12]. Cooperation of IDSs is still ongoing work [6].

We have proposed a completely integrated Security Operation Center (SOC), called SOCBox[1], in order to overcome the limitations of IDS. The SOCBox gathers data from a wide range of sources (IDS, firewall, router, workstation, etc.) and therefore has a global view of the network. Its analysis engine can then correlate all messages generated by all the network components and find patterns of intrusion.
To measure the detection capabilities and performance of the SOCBox, an evaluation has been performed with Snort [19] as a baseline. This evaluation has taken place in two different but complementary environments: a real Internet Service Provider network and our laboratory.

The rest of the paper is structured as follows: Section 2 designs the global architecture of the SOCBox. We then focus on the collection and the analysis of the data generated by sensors in sections 3 and 4. In Section 5, we focus on the SOCBox evaluation and we provide details about the experimentation. Section 6 summarizes the main results, presents our conclusions and describe further research to be performed in the SOCBox design.

[1]This project was partially funded by a CAPM, CRFC, government and EU programme under the STIC pole.

2 Global architecture

The SOCBox implements the different type of boxes defined for network intrusion detection systems in [13]. However, beside the pure technical aspects involved in such an implementation, it is necessary to consider the supervision of an IT infrastructure as a full operational project. We will thus follow the functional steps of such a project in order to describe both the purpose and the concepts of selected parts of the architecture described in figure 1.

2.1 Data acquisition

Before setting up sensors and designing any correlation or analysis rule, it is necessary to evaluate the overall security level of the IT infrastructure to be supervised. This will make it possible to determine if an intrusion path may effectively lead to an intrusion on the target system and the criticality associated to such an intrusion attempt.

Another point to be defined is the security policy, mostly in terms of access rights, permitted operations, etc.

2.1.1 Vulnerability database

The vulnerability database holds information about security breaches and insecure behavior that would either impact the overall security level or that could be exploited by an attacker in order to perform an intrusion. The database format must make it possible to include the following types of vulnerabilities :

- Structural vulnerabilities, i.e. vulnerabilities internal to specific software such as a buffer overflow, format string, race conditions, etc.
- Functional vulnerabilities, depending on configuration, operational behavior, users, etc. These vulnerabilities differ from the previous ones as they deeply depend on the environment in which they lie.
- Topology-based vulnerabilities, including networking impact on intrusions and their consequences.

2.1.2 Security policy

The next step of the supervised system inventory is an organizational one and, more specifically, a review of security policy aspects

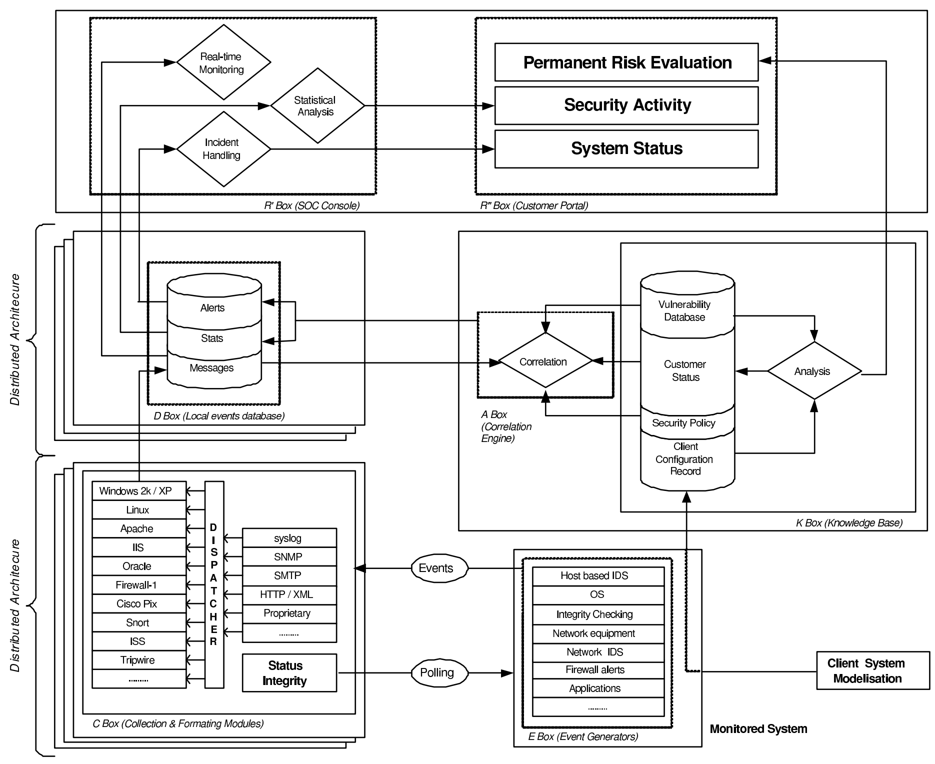


Figure 1: Global architecture of a SOC

that would impact either event generation and/or the reaction-reporting processes.

It is clear that the two major aspects of security policy that need to be reviewed are authorization and testing/audit procedures. These two aspects will provide information concerning behavior that sensors would detect. Events generated (administrator login, portscans, etc.) will then be marked as matching with security policy criteria. Others will be analyzed as possible part of an intrusion attempt.

This information is stored in the Knowledge Base.

2.1.3 Status evaluation

The last part of the Knowledge Base is a detailed security level evaluation of the systems to be monitored. The objective is to process such an evaluation through an analyzing engine capable of integrating the three kinds of vulnerabilities as seen above, as well as security policy constraints. The engine should provide a list of vulnerabilities each system is exposed to, the relative impact of each vulnerability and intrusion paths leading to the activation of "inactive" vulnerabilities.

In order to be reliable, such an evaluation must be re-generated each time a new vulnerability is found or one of the monitored systems is changed.

2.2 Data analysis and reporting

2.2.1 Structural and behavior-lead alerts

The main operations performed that generate alerts are the following: correlation, structural analysis, intrusion path analysis and behavior analysis. Correlation is a stand-alone operation leading to the creation of contexts against which further analysis will be made, in order to check if they match the characteristics of an intrusion attempt. Structural analysis may be compared to an advanced pattern matching process, used to determine if events stored within a certain context lead to a known intrusion path or to an attack tree [20]. Intrusion path analysis is the next step whose output provides information about the exposure of the target system to the intrusion attempt detected. Then, the behavior analysis integrates elements from the security policy in order to determine if the intrusion attempt is allowed or not.

The purpose of such operations is to generate alerts that not only match the structural path of intrusion (i.e. scan, fingerprinting, exploiting, backdooring and cleaning), but also take care of the security policy defined, as well as criticality of targets systems.

3 Collection and storage

3.1 Data collection

Collecting data from heterogeneous sources implies the setup of two kinds of agents: protocol and application. The former collects information from E Boxes, the latter parses information for storage in a "pseudo-standard" format. These two modules are connected by a dispatcher. Such an architecture allows high-availability and load-balancing systems to be set at any level into the architecture.

Figure 2 shows some architecture examples, based on the details provided below.

3.1.1 Protocol agents

Protocol agents are designed to receive information from specific transport protocols, such as syslog, snmp, smtp, html, etc. They act like server side applications and their only purpose is to listen to incoming connections from E Boxes and make collected data available to the dispatcher.

The simplicity of such agents make them easy to implement and maintain.

The raw format storage is usually a simple file, though direct transfer to the dispatcher through named pipes, sockets or shared memory ensures better performance.

From a security point of view, the most important point is to ensure the integrity of data collected by agents. Therefore, data is encapsulated into a secure tunnel.

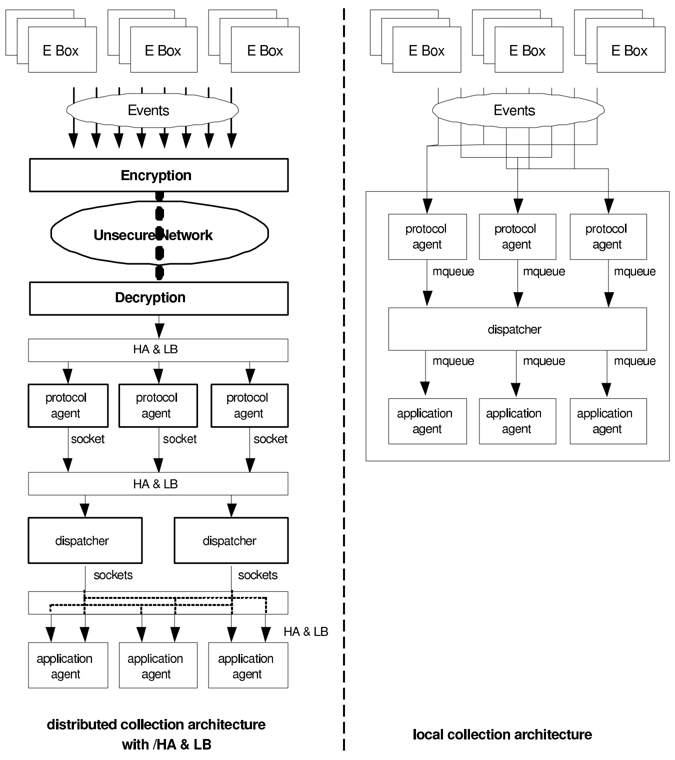


Figure 2: Collection Macro Architecture Examples

3.1.2 Dispatcher and application agents

The dispatcher's purpose is to determine the source-type of an incoming event and then forward the original message to the appropriate application agent. Once again, implementation is relatively trivial, once a specific pattern has been found for each source-type from which the data may be received.

Autonomous operations performed by the dispatcher are the following:

- listening to an incoming channel from protocol agents, such as socket, named pipe, system V message queue, etc.
- checking pattern matching against a patterns database that should be pre-loaded in memory for performances considerations.
- sending the original message to an E Box specific application agent through any suitable outgoing channel.

Application agents perform formatting of messages so that they match with the generic model of the message database.

Autonomous operations performed by application agents are:

- listening to an incoming channel from dispatchers, such as socket, named pipe, system V message queue etc.
- parsing the original message into standard fields.
- transmitting the formatted message to corresponding D Boxes.

3.2 Data formatting and storage

Two kinds of data have to be formatted in a "standard" manner (i.e. homogeneous and understandable by any module of the SOCBox): host entry and collected messages.

3.2.1 Host entry

The need for a standardized host data structure appears since:

- sensors may transmit host information in IP address format or FQDN (Full Qualified Domain Name) format.
- multi-homing techniques provide multiple IP addresses for the same physical system.
- virtual host techniques provide multiple FQDN for the same physical system.
- high availability and load balancing systems may hide multiple systems behind a single IP address or FQDN.

It appears that identifying a host either by its IP address or its FQDN is not reliable. What is more, in the never-ending need for performance, (reverse) DNS lookup cannot be performed for each new (IP address) FQDN detected in logs. It is then necessary to rely on a third-party ID, IP address and FQDN independent: the host token.

The data structure for storing host information follows the scheme given in Figure 3.

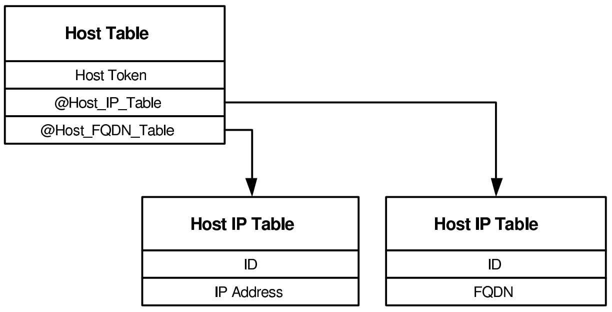


Figure 3: Host Entry Data Structure

3.2.2 Messages

Working on the data generated by the different types of equipment, transmitted via different protocols requires "standard" formatting. Although an effort has been made to define a worldwide standard with IDMEF [7], it appears that the XML bus used is too heavy and resources consuming, for our purposes of event correlation. However, a separate translation process must be implemented for IDMEF compliance. Relations between each structure are given in Figure 4.

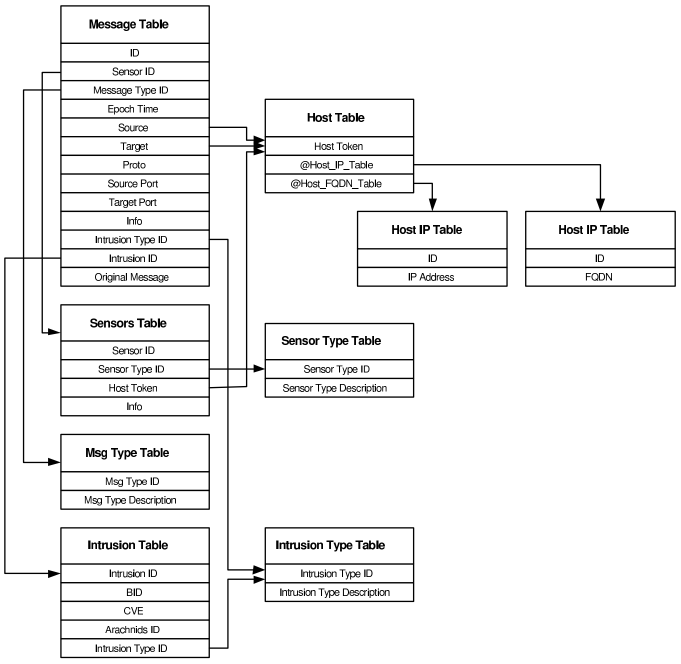


Figure 4: Formatted message definition structures

4 Correlation

4.1 Overview

4.1.1 Operating the correlation

The correlation's purpose is to analyze complex information sequences and produce simple, synthesized and accurate events. In order to generate such qualified events, five operations are to be performed:

- the first, obvious, operation is to identify duplicates and set a specific flag in order to keep the information and continue without the need keep multiple identical messages.
- sequence patterns matching is the most common operation performed by a correlation engine. Its purpose is to identify a sequence of messages which would be characteristic of an intrusion attempt. It makes it possible to identify on-going intrusion processes, as well as complex intrusion scenarios.
- time pattern matching is designed to include another important dimension in intrusion analysis: time. This is mainly used for context (see below) management, as well as slow and distributed intrusion processes.
- system exposure and criticality analysis, provide information about the target system's vulnerability to detected intrusion attempts. Indeed, it seems inappropriate to have the SOCBox generating alarms concerning an intrusion scenario based on a vulnerability that the target system is not exposed to. Another piece of information is the criticality of the intrusion i.e. its overall impact on the supervised system. This helps to manage the priorities in terms of reaction to multiple incidents.
- security policy matching, is a behavior-based filter that eliminates specific events if they match security policy criteria such as administrator login, identification processes and authorizations / restrictions.

A global overview of correlation operations is given in figure 5 below.

4.1.2 Introduction to contexts

The analysis defined above is based upon a specific structure called contexts. All correlation operations are performed against these structures. In simple terms, the definition of a context is the following: a container of formatted data matching a common criteria. Therefore, any message stored in the formatted message database is to be part of one or more contexts. Correlation operations will be done in parallel so that they can be run simultaneously on each context. Two kinds of context management approaches can be implemented. The first one is to define independent and distinct contexts. Each context will contain messages matching every criteria. We define such an architecture as an array of contexts. The second approach is a hierarchical one. Top level contexts matching a limited number of criteria are defined. Then sub-contexts, based on different criteria, are created and so on. This will be defined hereafter as context tree. As is to be expected, none of the preceding approaches meet all needs, be they in terms of performance or functionality. A mixed architecture will thus have to be defined.

4.2 Contexts

4.2.1 Context definition criteria

Defining context criteria is done according to security related events that the SOCBox must react to, even if they are distributed scanning operations, fingerprinting, massive exploit testing, account brute forcing, spamming and so on. A full functional architecture of contexts is given in figure 6.

The first obvious criteria is the attacking and attacked host's ID, which has to be standardized:

- source, defining source as a context creation criteria allows sweeps detection, identification of the systems used as attack relays or compromised by worms and targeted hack-proofing.
- target, contexts created by target criteria will provide information on scans (be they distributed, slow or "normal") and, should it even be noticed, intrusion attempts and system compromises.

Two arrays of context should then be defined, one with context matching sources, another matching targets. Each context of each array should then be considered as a top-level context for the context trees. The criteria to be matched by the smallest "branches" would be target ID (for contexts created by the source ID match) or source ID (for contexts created by the source ID match).

While working with trivial data, the protocols and the ports of the targeted systems should form the criteria for the next level of context "branches". This is mainly done in order to isolate single scanning operations from heavy repetitive attempts to compromise a system through a specific application. What is more, it helps to identify the various steps of an intrusion. Indeed one of the most common intrusion scenarios, is a wide portscan sweep followed by fingerprinting/version identification on open ports followed by an exploit launched on systems believed to be vulnerable.

In order to identify which type of message is stored, thus starting a more accurate analysis of messages, a next level of context generation is performed according to the intrusion type ID. The last "branch" of contexts contains specific intrusion ID, i.e. the characterization of each message. At this level we reach the atomic (except for duplicates) dimension of each message. This field refers to the Intrusion Table and will be responsible for the link between the correlation engine and system status information stored in the Knowledge Base.

4.2.2 Contexts structure

As any correlation operation is exclusively performed on contexts, it appears that their structure is probably one of the most important aspects of the SOCBox. The functional architecture is made up of an array of context trees. Each tree contains 4 levels of branches, as described in figure 6.

4.2.3 Contexts status

Another important characteristic of context is its status. We define three distinct statuses as detailed below:

- active, context matches specific criteria (usually based on time but could be any other criteria), which could be a characteristic of an on-going intrusion process. Typically, such a context appears to be under a heavy load from the arrival of new messages and its analysis by the correlation engine should be set to the highest possible priority.
- inactive, such a context either does not match "active" criteria or did not receive a specific closure code. This means that it is no longer analyzed by the correlation engine, but that it can be reactivated by the next message matching the same context criteria.
- closed, in this state a context is completed. Any new message matching this context criteria will create a new context.

Context status management is summarized in figure 7.

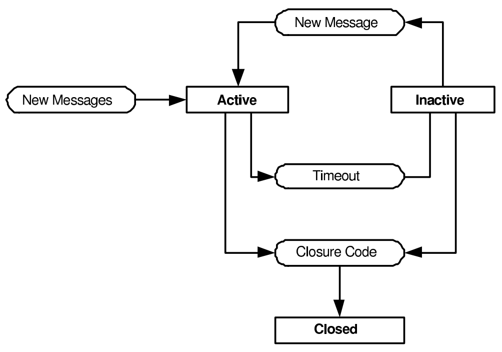


Figure 7: Context status management

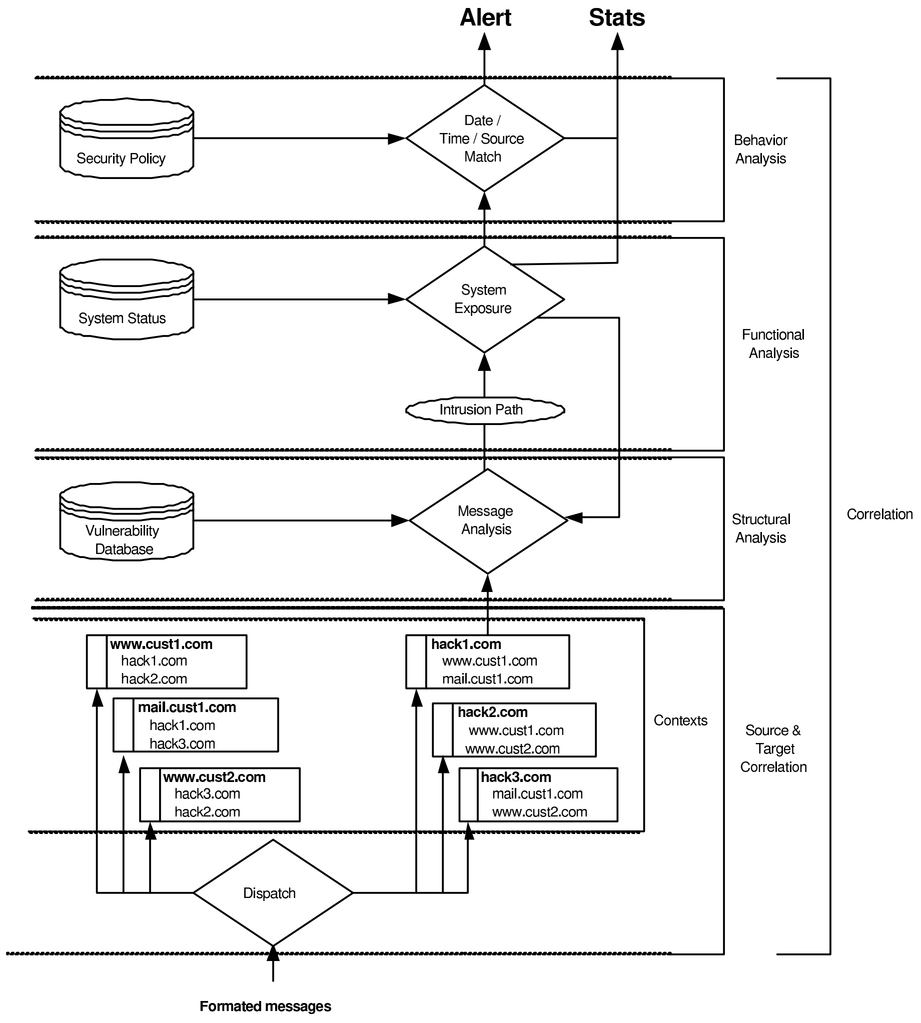


Figure 5: Main correlation operations

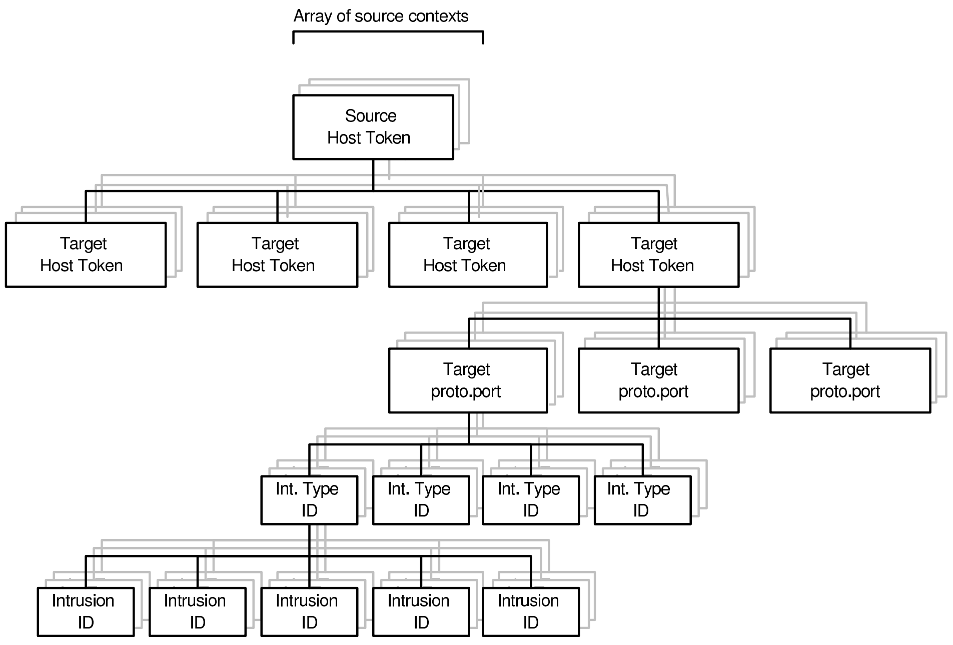


Figure 6: Contexts functional architecture

4.3 Analysis

4.3.1 Structural analysis

The purpose of structural analysis is to identify ongoing intrusion attempts, manage context inactivity status and context closure conditions. In simple terms, structural analysis is a set of operations performed by independent modules on each context. Each module is activated by a specific message and performs analysis using a "standard" semantic.

The output of the analysis modules is the result of several logical operations between autonomous conditions against fields of contexts. Figure 8 describes members of such operations.

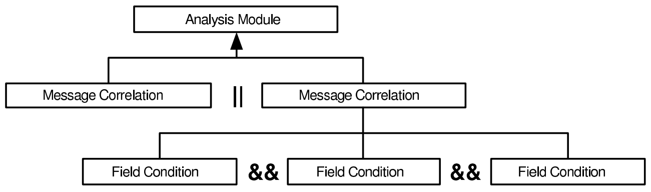


Figure 8: Analysis module structure

Field conditions have the following structure:

```
field operator <field | value> [!]
```

It appears that the power of structural analysis relies on the number of operators made available. However, the very structure of contexts provides embedded operations such as source, target, port correlation. This not only increases the number of "native" operators but also significantly improves the performances of structural analysis. The ! sign indicates that the field condition is to be matched in order to activate the module.

Two kinds of events can activate analysis modules: messages and time.

- messages, as described above, some field conditions must be matched in order to activate an analysis module. A header containing field conditions to be met, is then generated for each analysis module. Given the structure of analysis module, it appears that the header will be a set of logical OR operations, whose members will be field conditions that require the least amount of resources to be evaluated.
- time, the analysis module header may also contain timer information forcing the correlation to be evaluated. This is mainly used for context closure and time dependent intrusions detection such as (slow) portscans, brute forcing, etc.

4.3.2 Advanced correlation

Advanced correlation operations are performed in order to define the criticality of an intrusion attempt and evaluate if such an intrusion attempt is permitted according to the security policy.

The functional analysis step is performed in order to evaluate system exposure to the intrusion and the overall impact of such an intrusion on the supervised system. Once the structural analysis has provided information about an occurring intrusion attempt, a request is made to Customer Status part of the K Box. This request contains the Intrusion ID and the Host token of the target. The response provides the following pieces of information:

- criticality, is a value from an arbitrary scale, typically info-warning-minor-major-critical based.
- closure code, if the context is to be closed, usually because the target is not impacted by the intrusion attempt.
- message, a new formatted message to be appended to the actual context, that may activate additional analysis modules.

The purpose of this last analysis is to define if the attempts match the security policy. This is mainly used to manage access to accounts but can also be implemented in the case of pre-programmed audits, portscans, etc. In such a situation a closure code is sent to the context. Technically, this analysis is performed in exactly the same way as structural analysis i.e. via specific modules whose structure is loaded from the security policy part of the K Box.

5 The SOCBox evaluation

In this section, we evaluate the intrusion detection capabilities of the SocBox and its performance. The SocBox evaluation consists in running it in a real ISP network and to verify its capacity to manage events coming from heterogeneous platforms (routers and access servers, hardware and software firewalls, unix and linux servers, windows workstations, web and mail servers, a AAA server and other ISP applications). This test has taken place for a week. After that, some exploits were executed against the network to check the capacity of the SocBox to detect various classes of intrusions. Then, the ability of the SocBox to detect distributed intrusions is evaluated. After that, the clarity and the relevance of the SocBox reports are studied. Finally, performance evaluation take place in comparison with Snort.

5.1 The evaluation network design

The Socbox is evaluated in a real ISP network (Figure 9) which manages more than 50000 subscribers. This ISP network is composed by a core sub-net and several regional sub-nets.

5.2 Detection capabilities

5.2.1 Capabilities to manage heterogeneous platform events

For the Socbox be able to manage data coming from sensors, it is necessary to install log redirection towards it on sensors. To verify the SocBox capabilities to manage heterogeneous platform events, we run it in a real situation on a ISP network for a week. This showed multiple attempts at intrusion into the network servers (including the SocBox), in particular port scans, authentication attempts, brute force attacks, sql attacks, mail relay attempts, etc. These attacks are carried out on sensors running Solaris, Hp ux, Linux, Windows 2000, Cisco IOS, Pix OS and applications such as Bind, Tacacs+, Apache, etc.

5.2.2 Intrusions Detection Capability

In this part, some classes of attacks are launched against some critical sensors and the SocBox itself. The goal is to verify the intrusion detection capability of the SocBox. Some of the tests carried out are presented below :

Flood an pollution attacks	Detection	Comment
Flooding the SocBox with Harpoon [21] followed by a brute force attack (with THC-Hydra [24]) on a Cisco access server (Victim 3).	YES	The SocBox detects multiple "authentication failed" against the access server.
Flooding and polluting the SocBox with a random MAC address generator (Macof [23]) followed by a brute force attack (with THC-Hydra) on a router (Victim 3).	YES	The SocBox detects the brute force attack on the access server (multiple "authentication failed").

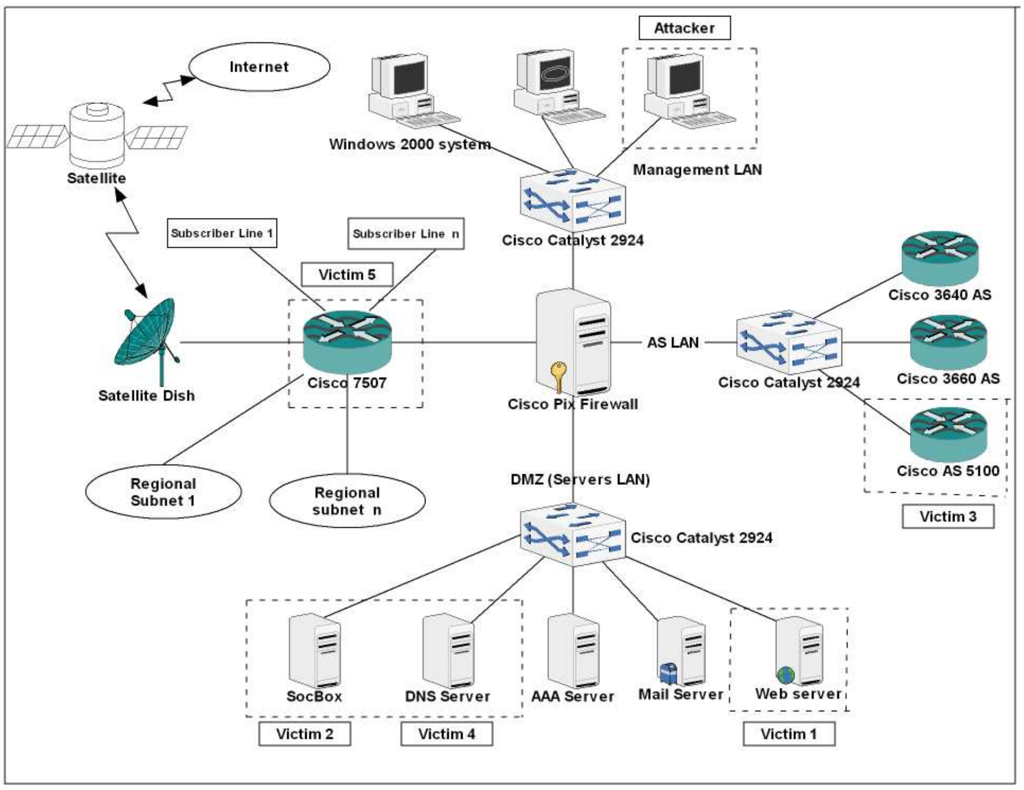


Figure 9: The ISP network used for the SocBox evaluation

Brute force and password cracking attack	Detection	Comment
Brute force attack against a router with THC-Hydra	YES	The attack is detected.
Attempt to crack password by John the Ripper [15]	YES	the attack is detected.

Scan and sniff	Detection	Comment
Scanning the network with nmap.	YES	The SocBox detects the scan (data are collected by the firewall sensor).
Sniffing the network with Dsniff [22].	NO	the SocBox is unable to detect the attack because it can not sniff on a network.

Fragmentation, insertion and camouflage attack	Detection	Comment
Fragrouter [16] attack on the backbone router (Victim 5) from a remote host.	YES	The intrusion is detected by the SocBox.
nmap with DECOY option (source IP camouflage).	YES	the SocBox detects the attack.

Offline detection capability	Detection	Comment
Replaying in the ISP site the DARPA 2000 [25] DDOS attack data set with TcpReplay [1].	YES	The SocBox generates alerts about the DDOS attacks.

Web attack	Detection	Comment
A Whisker [18] attack on a web server running Solaris and Apache 2 on the ISP site.	YES	The SocBox generates ”target identification” alerts.

Anomaly behavior intrusion detection evaluation	Detection	Comment
Attempt to connection at 9 pm to a windows 2000 server with a username authorized to connect only between 7 am and 8 pm.	YES	The SocBox generates a ”suspicious behavior” alarm.

Multi steps attacks	Detection	Comment
Lpr attack (lpr file1 (big file); rm file1; ln -s /etc/shadow file1)	YES	The intrusion is detected by the SocBox.

Attack against filtered ports and services	Detection	Comment
Executing a brute force attack with THC-Hydra on a router (Victim 5) having ssh, telnet, rlogin and rsh filtered.	YES	The intrusion is ignored by the SocBox because it can never succeed.
Executing a web server attack on the DNS server (Victim 4, which does not run a web server).	YES	The intrusion is ignored by the SocBox because it can never succeed.

As we can see, the SocBox is able to detect various classes of intrusions, suspicious behavior (defined by the security manager) and it can ignore events which generate useless alerts (attacks against non-vulnerable systems). It also appears that the more sensors

send their logs to the SocBox, the better its detection capability is. To improve its efficiency it is possible to associate with it an intrusion detection system based on a behavioral approach or a tool which can intercept and format network packets and redirect them towards it. Online exploits executions and the replay of DARPA 2000 data sets show that the Socbox can detect online and offline intrusions. The security manager also plays a major role in the SocBox detection efficiency; the more good rules of detection he writes, the more efficient the SocBox is. The security manager is also responsible for actions taken against intrusions.
In summary, we can say that the Socbox has proved its efficiency in detecting intrusions and in presenting the network security status clearly by using graphical representations.

5.2.3 Detection of distributed Intrusions

The evaluation of the aptitude of the SocBox for detecting distributed intrusions is described on Figure 10. The scenario of this attack is the following:
An attacker wants to hack a host (*Victim*) located on the ISP core sub-net and hosting information about subscribers. His idea is to gain access to *Victim* by brute force attack and to steal data about subscribers. *Victim* is secured and can be acceded only from special hosts in the Management LAN and in some regional sub-nets (for maintenance purpose). The attacker tries to compromise *Victim* and unfortunately for him, all his actions are refused. After further thought, he thinks that it would be easier for him to try to hack *Victim* from hosts located on the ISP network. He uses social engineering technique to know the name of the administrators of the ISP core and regional sub-nets; this can help to improve the username database of the brute forte attack tool. After several attempts at intrusion, he compromises three less secured hosts on the ISP network (one in the Management LAN and two in regional sub-nets). From these hosts, he initiates the attack, composed of the following actions:

- From *Attacker 1* located in a regional sub-net, he launches a quick scan (with nmap) to detect opened ports on *Victim*. He sees that ssh and mysql are opened on *Victim*.
- From *Attacker 2* located in the ISP core sub-net, he executes an OS Fingerprinting with Xprobe2. He see that *Victim* runs Solaris 8.
- From *Attacker 3* located in another regional sub-net, he launches a brute force attack (with THC-Hydra) against *Victim* to gain access to the mysql database.

This test shows that the SocBox can gather events and alerts coming from different sensors (Cisco Pix Firewall sensor detects the quick scan, Snort sensor detects the OS Fingerprinting, and logs of *Victim* permit to detect the brute force attack). Because these events have the same target and they take place approximatively in the same time, the SocBox matches them with the same context and generates a suspicious behavior alert. An alarm is also sent to the security manager for advanced investigation on *Attacker 1*, *Attacker 2* and *Attacker 3*. Investigation on these hosts shows that Attacker acceded them. Then, the security manager concludes that *Victim* is attacked by Attacker.
Without correlation of alerts, it would be impossible to detect this attack. The SocBox is thus able to correlate alerts coming from divers sources (firewalls, IDS, hosts, etc.) to generate a single alert. Many NIDS can not detect this kind of multi events intrusion.

5.3 Performance Evaluation

At this stage, we check the aptitude of the SocBox to handle high bandwidth traffic and its ability to detect intrusions when a massive attack occurs. We uses D-ITG [4] and IP-Traffic [26] to generate traffic in this test. The same tests are apply to Snort for comparison purpose. In spite of the fact that Snort isn't a Soc and the SocBox isn't an NIDS (the SocBox is much more than an NIDS because it has a global view of a network security), the comparison between Snort and the SocBox is justified in this test: Both monitor the security of a unique host and they generate alerts only about attacks on this host.

5.3.1 Evaluation of the SocBox maximum processing capacity

A victim host (which sends its log to the SocBox via syslog) is flooded and attacked (Figure 11(a)). Then, we observe the SocBox behavior. The Socbox host characteristics are: Pentium III, 450 MHz, 256MB of RAM. The same tests are carried out with Snort installed on a host which have the same characteristics (Figure 11(b)). The tests are summed up in the following tables.

Action	The SocBox reaction	Snort reaction
Launching a Whisker attack on a victim running Apache.	The Socbox generates "Target identification" alerts.	Snort generates "WEB-MISC whisker" alerts.
Flooding the SocBox and Snort with 10^5 packets of 500bytes each second for 15mins followed by a Whisker attack.	The SocBox generates alerts about the whisker attack (the Socbox host uses 245 MB of RAM).	Snort generates alerts about the whisker attack and it ran slowly (around 250 MB of RAM is used).
Flooding the SocBox and Snort with 10^6 packets of 10 bytes each second for 15mins followed by a Whisker attack.	The SocBox runs slowly and it detects the Whisker attack (the SocBox host uses more than 245 MB of RAM).	Snort detects the Whisker attack and it runs too slowly (around 250 MB of RAM is used).
Flooding the SocBox and Snort with $1,2 * 10^6$ packets of 10 bytes each second for 15mins followed by a Whisker attack.	The SocBox detects the Whisker attack (around 250MB of RAM is used by the SocBox host). It runs slowly.	Snort host has not enough memory to continue (all the memory is used up).
Flooding the SocBox and Snort with $1,4 * 10^6$ packets of 10 bytes each second for 15mins followed by a Whisker attack.	The SocBox host has not enough memory.	

After that, ping with large packet flood is carried out against both the Socbox and Snort, followed by a Whisker attack against the victim host. The goal is to observe the behavior of the SocBox and Snort under a strong attack. The victim host characteristics are: Pentium III, 450 MHz, 256 MB of RAM.

- *Action*: Sending 20 millions ($2 * 10^7$) Ping with 50000 bytes each one (time between 2 Pings = 0) to the Victim host (via IP-Traffic), followed by a Whisker attack.
- *The SocBox behavior*: Up to $1,8*10^6$ Ping, the Socbox is able to detect the Whisker attack. At $1,9 * 10^6$ Ping the SocBox host is broken down and is unable to detect the Whisker attack.
- *Snort behavior*: Snort generates too many alerts about the Ping (10^5 Ping generate 100576 alerts, including 50283 Large ICMP packets detected). At 10^6 Ping, Snort generates 4GB of dumped data and is unable to generate alerts.

5.3.2 Comments

This test proves that the SocBox is able to detect intrusions under a high traffic or under a massive attack. Under a massive attack the SocBox uses less resources (CPU and memory) than Snort and

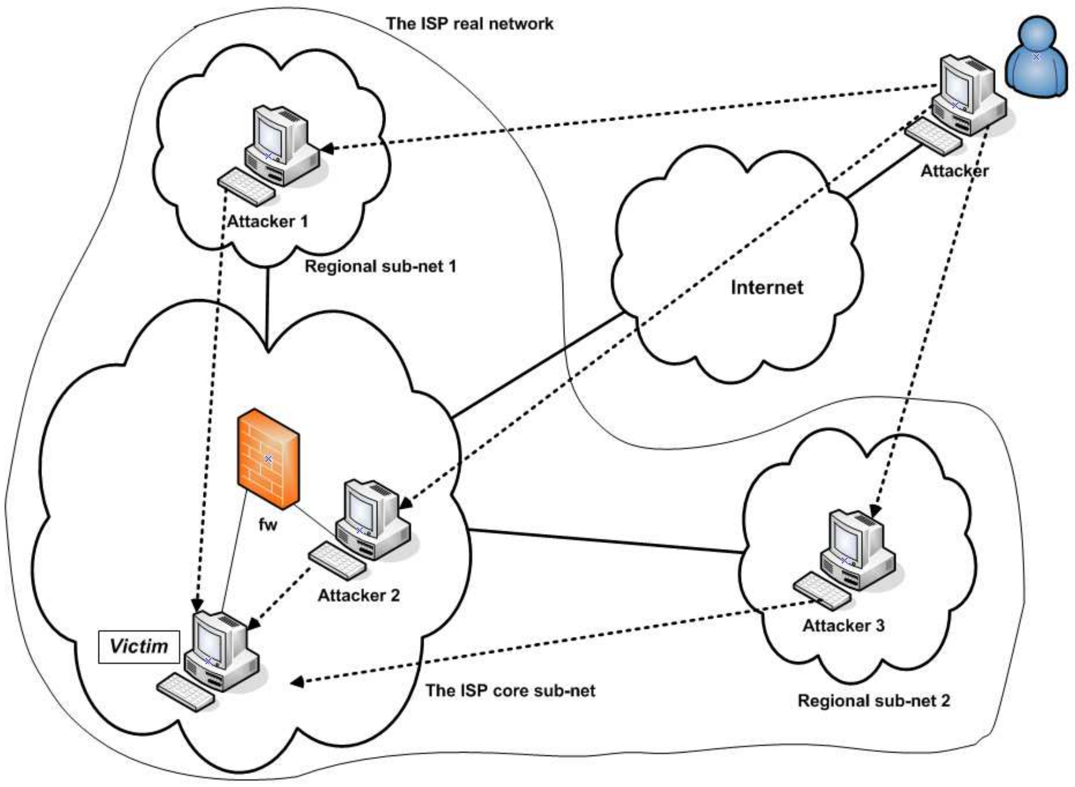


Figure 10: Evaluation of the SocBox aptitude for detecting distributed intrusions

has better performance. It also generates far fewer alerts than Snort and is able to compact similar alerts to generate one only. Moreover, the SocBox only records events that match security rules defined by the security manager. This allows a fine management of the SocBox hard disk space. On the whole, the detection performance of the SocBox is closely related to the following factors:

- The capacity of the security manager to define good detection rules.
- The number of active sensors in the network (the more sensors we have , the more powerful the SocBox is).
- The number of application protocols activated on each sensor.
- The sensors and the SocBox host characteristics (particularly the CPU and RAM).

Concerning Snort, the more powerful its host is (CPU, RAM and hard disk), the better the detection performance is.
The Socbox and Snort performance, memory usage and hard disk usage during the Ping test are shown on the following graphs (Figures 12, 13, 14, 15)

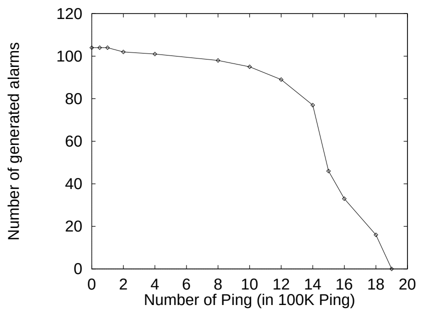


Figure 12: SocBox Ping test result

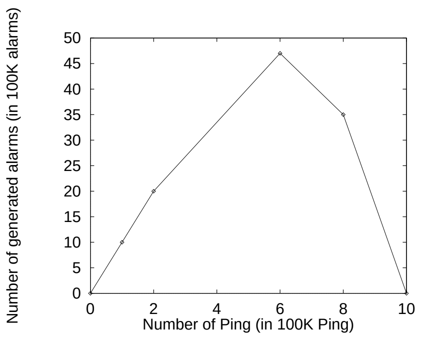


Figure 13: Snort Ping test result

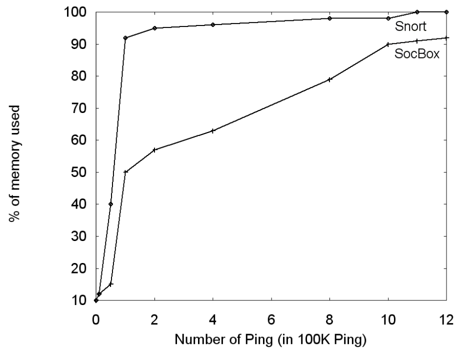
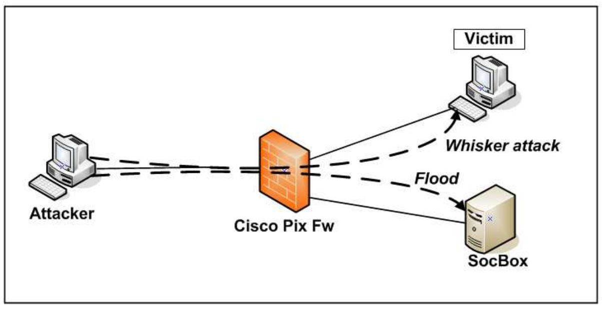
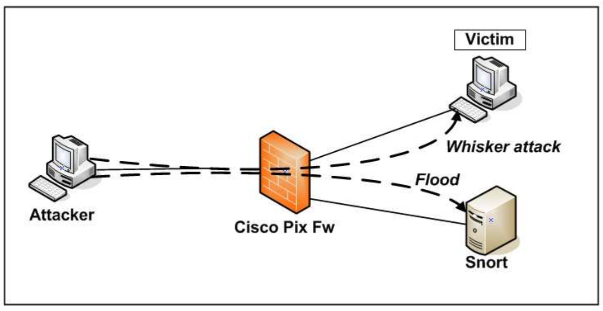


Figure 14: Snort and the Socbox memory usage during Ping test



(a) The SocBox performance evaluation



(b) Snort performance evaluation

Figure 11: The SocBox and Snort performance evaluation networks

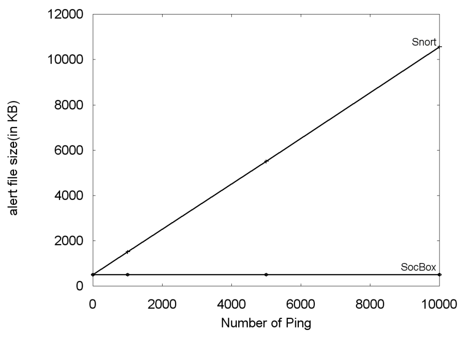


Figure 15: Snort and Socbox alert file (in KB) during the Ping test

5.3.3 Clarity of the SocBox reports

The Socbox has a reporting module which allows the generation of reports in HTML and PDF format. These reports can be generated in a scheduled manner or on-the-fly. On-the-fly reports are generated to obtain specific metrics or to troubleshoot when an incident occurs. Scheduled reports are automatically generated at regular intervals and are transmitted by email to one or more recipients and stored in a database. The SocBox is able to generate 4 types of report:

- The standard report (user report) gives information about the starting time and ending time of attempts at intrusions as well as their sources and targets. This kind of report is generally used to have a global view of the intrusion activity in a network.
- The operational report classifies events by intrusion source, target, date and type. It helps the security manager to improve his network security or to troubleshoot.
- The strategic report gives a macroscopic view of a network security (number of events related to intrusions, main intrusion types, main sources and targets). It is mainly used by decision-makers to justify the investments.
- A firewall report is a dedicated packet filtering summary report. It gives information about the number of events, the services most often targeted, operations on firewall, the intrusion starting and ending times, the sources and targets of intrusions. It can help to troubleshoot or to improve the firewall security.

Reports generated by the SocBox give much information about the network security, in particular the most active days, the most attempted intrusions, a temporal analysis of these intrusions, their principal sources and targets. According to the type of report, there will be more or less comment on each topic. Graphs enrich the reports, giving them more clearness and legibility. To have more details about intrusions, the report users could refer to the intrusion description provided by the SocBox. We also advise the security managers to consult the description of the intrusions on Snort website because it gives more details.

5.3.4 Relevance of the SocBox reports

The SocBox records only events which match security rules defined by the security manager. A visualization of the recorded events and generated alerts gives information about the intrusion sources, their targets, their description as well as their status (success or failure). It is also possible to see intrusions attempted into a given target or those executed from a given source. This method allows a personalized approach of the analysis of intrusions and gives the security manager a global view of the security of his network. He will thus be more reactive when an intrusion occurs.

5.4 Well-known intrusion detection systems evaluations

Various papers coming from both academy and industry laboratories and related to intrusion detection system evaluation have been published. Some industrial evaluations are biased because the tests are not always done in an objective way; the behavior of IDSs are adapted to the data sets and some tests are carried out without baseline. In the following paragraphs, we will present some well-known intrusion detection systems evaluations, coming primarily from academy laboratories.

5.4.1 MIT Lincoln Labs evaluation

Sponsored by DARPA in 2000, this evaluation [11] is one of the best-known intrusion detection tests. This evaluation uses a testbed which generates live background traffic containing hundreds of users and thousands of hosts. More than 200 instances of 58 attack types are embedded in 7 weeks' training data and 2 weeks' test data. The goal is to evaluate the efficiency for more than 18 research IDSs to detect unknown attacks without first training on instances of these attacks. Automated attacks are launched against a router and hosts running Unix/Linux and Windows NT. Attacks include Dos, user to root, probe, remote to local attacks. The drawback of this evaluation is the lack of baseline.

5.4.2 The UCAD evaluation

In this evaluation [17], automated attacks using TELNET, FTP and RLOGIN sessions were executed to evaluate a NIDS called Network Security Monitor (NSM) [10]. Scripts of normal and intrusion sessions were executed to verify the ability of the NSM to distinguish between suspicious behavior and normal one. Its ability to handle high bandwidth traffic was also evaluated. This evaluation has shown that NSM was unable to detect intrusions under high CPU load. A similar IDS evaluation [9] was performed by IBM Zurich in 1998 to improve IDS systems designed to detect intrusions into FTP servers.

5.4.3 MITRE evaluation

The goal of this evaluation [2] is to investigate the characteristics and capabilities of network-based intrusion detection systems. In this evaluation, 7 IDSs were tested using a two-phase approach. The first phase consisted in launching simple attacks using tools such as Satan [8]. The goal was to give IDS operators and attackers an opportunity to familiarize themselves with the IDSs. In the second phase, simulation of attacks took place. The IDSs were evaluated according to criteria such as reporting capabilities, off-line detection capabilities, real-time alert or response capabilities.

5.4.4 The NSS Group evaluation

In this evaluation [14], 15 commercial IDS and Snort were compared using 18 or 66 commonly available exploits such as Dos, DDos, ports scan, Trojans, FTP, HTTP or IDS evasion technique attacks. These systems were evaluated according to the following criteria: their ease of installation and configuration, their architecture, the type of reports and analysis provided. Only attacks reported in "as straightforward and clear a manner as possible" were supposed to be detected. In this evaluation, attack detection rates are difficult to compare with the other IDS evaluations because the simple detection of an intrusion is not sufficient; each generated alert must be clearly labeled to be taken into account.

6 Conclusion

Intrusions are clearly taking place and thus there is a need for operational supervision systems today. Experience shows that a pragmatic approach needs to be taken in order to implement a professional SOC that can provide reliable results. The SOCBox is our response to these new threats.

During its evaluation, the Socbox proved that it is a powerful tool giving the cartography of network security in a graphical and ergonomic way. It generates clear reports including graphs for helping the security managers better and has an interface for security alert consulting. It also has the ability to compact similar alerts to facilitate the legibility of the generated reports; this can be a great advantage during a troubleshooting operation for example. Moreover, the SocBox does not need a powerful host: its detection performance is closely linked to the capacity of the sensors to send it their logs.
However, some research are still to be conducted to improve our architecture: functionalities of the SOCBox have to be distributed on different network components. This will ensure the system scalability and messages will be better processed.

References

[1] T. Aaron and B. Matt. Tcpreplay tool (2.3). http://tcpreplay.sourceforge.net, 2005.

[2] J. Aguire and W. Hill. Intrusion detection fly-off: Implications for the united states navy. *in MITRE Technical Report 1997*, 1997.

[3] J.P. Anderson. Computer security threat monitoring and surveillance. Technical report, April 1980.

[4] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre. D-ITG a distributed internet traffic generator. http://www.grid.unina.it/software/itg/, 2004.

[5] F. Cuppens. Managing alerts in a multi-intrusion detection environment. In *17th Annual Computer Security Applications Conference*, New-Orleans, December 2001.

[6] F. Cuppens and A. Miege. Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Research in Security and Privacy*, Mai 2002.

[7] D. Curry and H. Debar. Intrusion detection message exchange format data model and extensible markup language (xml) document type definition. Technical report, IETF Intrusion Detection Working Group, January 2003.

[8] F. Dan and V. Wietse. SATAN : Security administrator 's tool for analyzing networks., 1995.

[9] H. Debar, D. Morin, and A. Wespi. Reference audit information generation for intrusion detection systems. In *Proceedings of IFIPSEC 98*, pages 405–417, 1998.

[10] T. Heberlein, V. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A network security monitor. In *IEEE Symposium on Research in Security and Privacy*, pages 296–304, 1990.

[11] R. Lippman, J. W. Haines, D. J. Fried, J. Korba, and D. Kumar. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In *3th symposium on Recent Advances in Intrusion Detection 2000*, pages 162–182, 2000.

[12] P. G. Neumann and P. A. Porras. Experience with EMERALD to date. In *First USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 73–80, Santa Clara, California, apr 1999.

[13] Stephen Northcutt and Judy Novak. *Network Intrusion Detection*. ISBN: 0-73571-265-4. New Riders, third edition edition, 2002. September.

[14] NSS-Group. Intrusion detection systems group tests (edition 2). http://www.nss.co.uk/ids, 2001.

[15] Openwall-Project. John the ripper password cracker (1.7). http://www.openwall.com/john/, 2006.

[16] T. H. Ptacek and T. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc., Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6, 1998.

[17] N. Puketza, M. Chung, R. Olsson, and B. Mukherjee. A software platform for testing intrusion detection systems. *IEEE Software*, 14(5):43–51, 1997.

[18] R. F. Puppy. A look at whisker's anti-ids tactics. http://www.wiretrip.net/rfp/txt/whiskerids.html, 2003.

[19] M. Roesch. Snort - lightweight intrusion detection for networks. In *proceedings of USENIX Large Installation Systems Administration Conference (LISA) 99*, 1999.

[20] B. Schneier. Attacks trees. *Dr. Dobb*, 1999.

[21] J. Sommers. Harpoon: A flow-level traffic generator http://www.cs.wisc.edu/ jsommers/harpoon/, 2005.

[22] D. Song. Dsniff 2.3: A collection of tools for network auditing and penetration testing http://www.monkey.org/ dugsong/dsniff/, 2001.

[23] D. Song. Macof - flood a switched lan with random mac addresses http://www.groar.org/trad/dsniff/dsniff-2.3/english-txt/macof.8.txt, 2001.

[24] THC. The hacker's choice, thc releases, thc-hydra v5.2. http://www.thc.org/releases.php, 2006.

[25] M. Zissman. Darpa intrusion detection evaluation data sets. http://www.ll.mit.edu/ist/ideval/, 2002.

[26] Zti-Telecom. IP traffic (2.3), a test and mesure tool. http://www.zti-telecom.com/fr/pages/iptraffic-test-measure.htm, 2005.

Author Biographies

Abdoul Karim Ganame is a PhD student at University of Franche-Comte, France. He received the Master degree in Computer Sciences from the Institut National des Sciences Appliquees (INSA), Lyon, France. His research interests are computer security and intrusion detection in wide networks and grids.

Julien Bourgeois is an assistant professor at University of Franche-Comte, France. During his PhD, he has developed an environment called ChronosMix which predicts performances for distributed applications executed on heterogeneous networks of workstations. His research topics include multimedia, P2P computing, performance prediction, Processor modeling, computer security and intrusion detection.

Renaud Bidou has been working in the field of IT security for about 10 years. He first performed consulting missions for telcos, pen-tests and post-mortem audits, and designed several security architectures. Currently, he is working with Radware Inc. as the security expert for Europe. He is also a PhD student at University of Franche-Comte.

Francois Spies is a Professor of computer sciences at University of Franche-Comte, France. His research and teaching interests include Resources Allocation in Distributed Systems, Performance evaluation of portable parallel systems, multimedia, computer security and intrusion detection.