COMPUTERS & SECURITY XXX (2008) 1–18

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

A global security architecture for intrusion detection on computer networks

Abdoul Karim Ganame, Julien Bourgeois, Renaud Bidou, Francois Spies*

LIFC, Universite de Franche Comte, 4 Place Tarradin, 25200 Montbeliard, France

ARTICLE INFO

Article history:
Received 1 December 2006
Received in revised form
13 September 2007
Accepted 10 March 2008
Published online ■

Keywords:
IDS
Distributed intrusion detection
SOC
Network security
Global view

ABSTRACT

Detecting all kinds of intrusions efficiently requires a global view of the monitored network. Built to increase the security of computer networks, traditional IDS's are unfortunately unable to give a global view of the security of a network. To overcome this situation, we are developing a distributed SOC (Security Operation Center) which is able to detect attacks occurring simultaneously on several sites in a network and to give a global view of the security of that network. In this article, we present the global architecture of our system, called DSOC as well as several methods used to test its accuracy and performance.

1. Introduction

Two aspects are to be taken into account when ensuring network security: protection and supervision. Protection is composed of hardware, software and security policies that must be pursued. Even the best protection is always vulnerable to attacks due to unknown security bugs. The network configuration is constantly changing, thus increasing the possibility of creating security holes. In order to help security administrators, IDS have been developed, but they have several drawbacks: insufficient detection rates, too many intrusions detected or missed (Cuppens, 2001). Moreover, basic IDSs have insufficient information to detect complex intrusions like distributed or coordinated attacks.

On the basis of our latest research work which presented a centralized Security Operation Center (SOC) called SOCBox (Ganame et al., 2006), we are developing a distributed one named DSOC to overcome the limitations of centralized systems on intrusion detection on wide networks.

The SOCBox is quite different from an IDS. It gathers data from a wide range of sources (IDS, IPS, firewall, router, workstation, etc.) and therefore has a global view of the security of the network. Its analysis engine is able to correlate all messages generated by all the network components and find patterns of intrusion.

With DSOC, on any network site, a local detection engine analyzes the data collected by one or several collection boxes to find intrusion patterns. Afterwards, all the generated alerts are processed by a global intrusion detection engine to find more complex intrusions and to give a global view of the network security.

The rest of the paper is structured as follows: Section 2 describes some drawbacks of centralized security systems collecting data from remote sites and explains why distributed systems are needed. Section 3 focuses on the description

* *Corresponding author.*
E-mail address: ganame@lifc.univ-fcomte.fr (A. Karim Ganame).
0167-4048/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2008.03.004

of the global architecture of our distributed system (the DSOC). In Section 4, presents our experiments results. Section 5 will be devoted to the related work and will be followed by a conclusion.

2. Needs for a distributed SOC

Our centralized SOCBox has some limitations: in addition to the fact that it was designed with a unique analyzer, giving a single point of failure, its performance evaluation (Ganame et al., 2006a). Ganame et al. (2006b) showed that its detection capability can decrease under a strong attack or with high traffic. Another drawback of the SOCBox is its inability to detect intrusions on a remote site in case of failure of the communication link between the site where the SOCBox is located and a remote site. Moreover, when the monitored network grows and we have to add several new sensors, the performance of the SOCBox can decrease.

These problems are common to all centralized security systems collecting data from remote sites.

2.1. *Single point of failure*

One of the principal drawbacks of a centralized SOC is its centralized architecture which induces a single point of failure. This situation increases the probability of denial of service which can decrease the global performance of the SOC.

In order to ensure continuous operation, two or more SOCs can be used in failover mode, but that does not ever resolve the scalability problem or the decreasing performance under strong attacks.

2.2. *Communication link breaking in a multi-site network*

For a centralized SOC to be able to manage sensors located on several sites, it is necessary to install VPN links between these sites and the one where the centralized SOC is located. One of the major drawbacks of this approach is that when a strong attack occurs on a site, the redirection of the logs (or the alerts) towards the centralized SOC can generate a large data flow which can break the communication link between the concerned site and the SOC. This prevents intrusion detection on the attacked site.

The scenario to verify the behavior of a centralized SOC when it manages several sites and when a strong attack occurs on one of them is described below. We named this attack "**isolation attack of a site**". A centralized SOC (the SOCBox in this test) manages the security of some critical sensors located on a network composed of 3 sites A, B and C (Fig. 1).

The SOCBox is installed on site A and the sensors located on the other sites send their logs to it through a VPN link.

After a scan of the network, a hacker sees open ports on sensors located on site B and decides to hack this site (his purpose is to steal data). Using a traffic generator, he floods the sensors with data containing the signatures of real attacks in order to break down any possible IDS installed on the site. The goal of this operation is to camouflage his intrusion.

After a few minutes's flood, the hacker launches an attack on site B. He compromises a sensor and steals data. After that, he erases his actions on the logs of the compromised sensor.

During this attack, when the flood occurs, the attacked sensors generate large quantity of logs which are redirected towards the SOCBox. Due to the high data flow sent towards

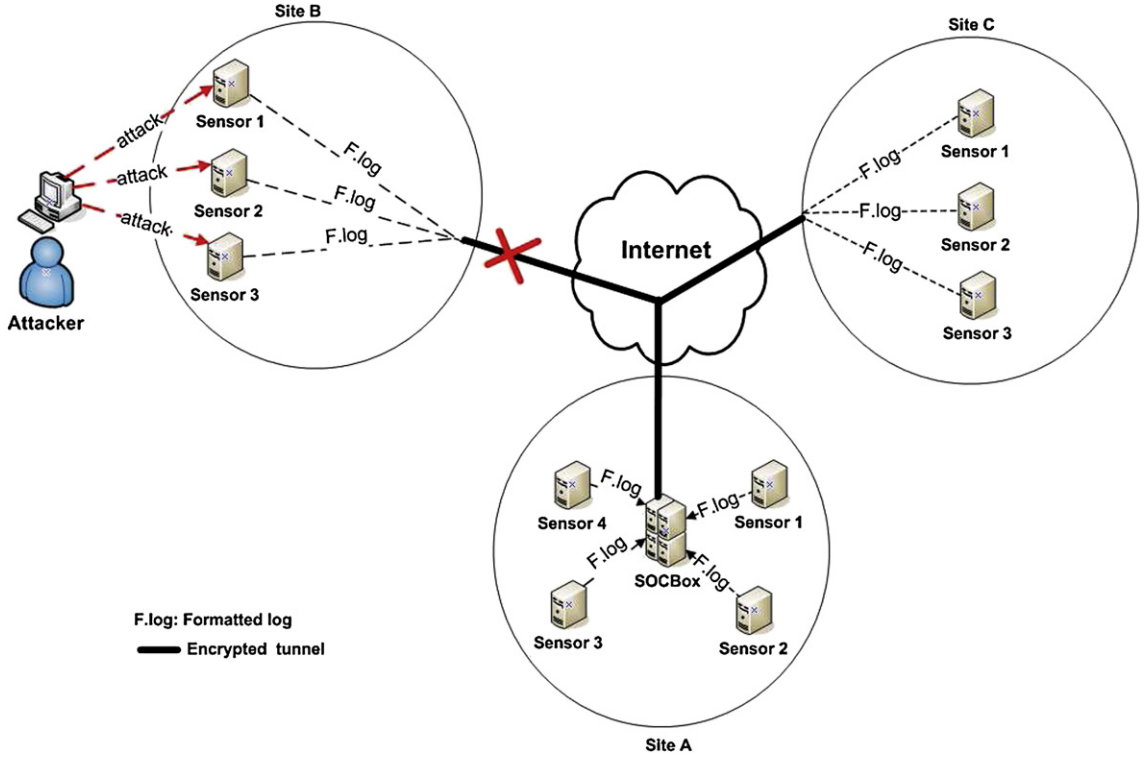

Fig. 1 – Isolation attack of a site managed by a centralized SOCBox.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

the SOCBox, the VPN link is saturated and the communication link between site B and site A goes down.

The network administrator notices the interruption of the VPN link and restores it. But, because the SOCBox does not receive all the logs coming from the compromised site, it cannot determine if the intrusion in site B was successful or not. Thus, the security administrator cannot conclude that data is stolen.

2.3. *Limitation of a centralized SOC when operating on a single site*

The goal of this test is to check the limitation of centralized SOCs when they manage a single site where a strong attack occurs. In other words, we try to answer the question "Is a centralized SOC able to continue to detect intrusions when it receives a high data flow?".

We use the SOCBox as a centralized SOC in this test which consists in flooding some sensors on a network with a high data flow composed of Pings with large ICMP data (50000 bytes each one) and to initiate a Nikto attack (Puppy, 2006) against one of the sensors (a web server). The goal is to check if the SOCBox is able to detect the Nikto intrusion.

The experiment is carried out by sending data to the SOCBox using two methods, namely:

1 Sending log to the SOCBox via a network packet sniffer (Snort in this test).
2 Sending log to the SOCBox via syslog.

The host characteristics are

Victims: PIII, 450 MHz, 256 MB of RAM
The SOCBox and Snort: PIII, 450 MHz, 256 MB of RAM
Attacker: PIV, 1.73 GHz, 512 MB of RAM.

2.3.1. *Using a network packet sniffer (Snort) to send data to the SOCBox*

When sniffing data in the network (Fig. 2), Snort generates a great quantity of log and sends it to the SOCBox.

At 1 million Pings, Snort generates 4 GB of log and is unable to send the log to the SOCBox (the Snort host has not enough memory to continue).

2.3.2. *Using syslog to send data to the SOCBox*

When the SOCBox receives events coming from the sensors (Fig. 3), it automatically analyzes them and records them on the hard disk only when they match security rules defined by the security administrator. Because we configured the SOCBox so that it ignores the Pings, it only records events about the Nikto attack. Thus, from 0 to 1.8 million Pings, the SOCBox records 500 kB of data each time the Nikto attack is launched. Over 1.8 millions Pings, the SOCBox is unable to detect the intrusions, due to the fact that it uses a great quantity of memory.

2.3.3. *Memory usage*

The memory usage of the SOCBox and Snort during this test is shown in Fig. 4. We notice that when a network packet sniffer is used to forward logs to the SOCBox, the performance of the SOCBox is closely related to the capacity of this packet sniffer to forward the logs. When this packet sniffer is down, the SOCBox cannot detect anything.

2.4. *Bandwidth usage when the SOCBox gathers data from remote sensors*

In this test (Figs. 5 and 7), we successively flood one sensor, two sensors, and four sensors installed on a remote site which forward their logs to a centralized SOCBox located on another site. The goal is to measure the bandwidth usage induced by

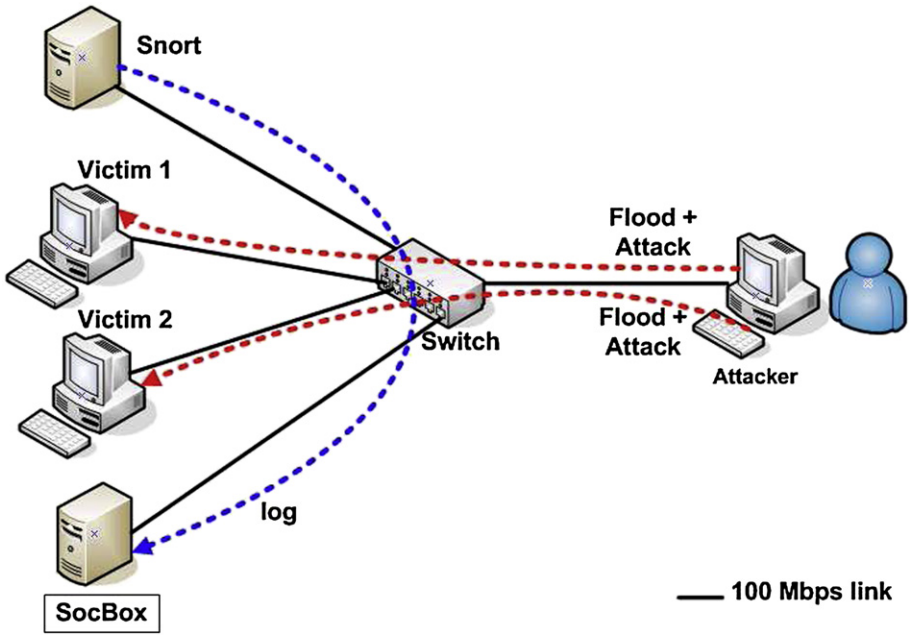

Fig. 2 – Using a network packet sniffer (Snort) to send data to the SOCBox.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

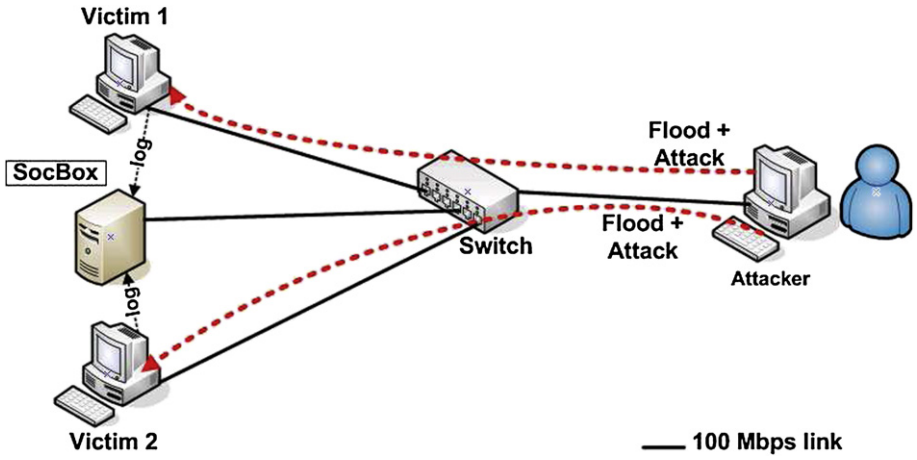

Fig. 3 – Using syslog to send data to the SOCBox.

data redirection towards the SOCBox when a strong attack occurs.

In the first part of the test, we flood a sensor by sending it 2 million Pings, each with packets of 1460 bytes (via IP-Traffic, Zti-Telecom, 2005). Then we observe the bandwidth usage when logs are forwarded to the SOCBox located on the remote site. The two sites are connected using two Cisco 2600 routers with a 25 Mbps link.

In the second part of this test, two sensors installed on the same site are flooded simultaneously by sending each 2 million Pings with 1460 byte packets. These sensors forward their logs to the SOCBox located on another site. In the third part of this test, 4 sensors are used in the same conditions as in the second part of the test.

2.4.1. *The sensors send their logs to the remote SOCBox via syslog*

When flooding the sensors for 30 min (Fig. 5), we notice a stabilization of the bandwidth usage around 12 Mbps (with 1 sensor) and 24.8 Mbps (with 2 sensors). With 4 sensors, the bandwidth is saturated very quickly after 5 min (Fig. 6) and it causes losses of packets in the first place, and the breaking of the communication link between the two sites in the second place.

2.4.2. *A network packet sniffer sends log to the SOCBox*

During this test (Fig. 7), we flooded two sensors with Pings (50,000 bytes each for 30 min). The logs are collected by Snort which forwards them to the SOCBox located on another site. Then we measure the bandwidth usage. The two sites are connected using two Cisco 2600 routers with a 25 Mbps link.

The bandwidth usage during the log transfer towards the SOCBox is stabilized around 730 kbps (Fig. 8(a)). Data (153 MB) are transferred for 30 min (Fig. 8(b)).

With this transfer mode, when a massive attack occurs on a site (generating too many logs), it is impossible to have a real-time view of the security of the concerned site if the SOCBox is located on another site. This is due to the fact that data transfers to the SOCBox take a long time. Thus, the major drawback of this log transfer mode is the time induced by the redirection of log to the remote SOCBox. This can take from a few minutes to several hours, which is unacceptable in a network security supervision.

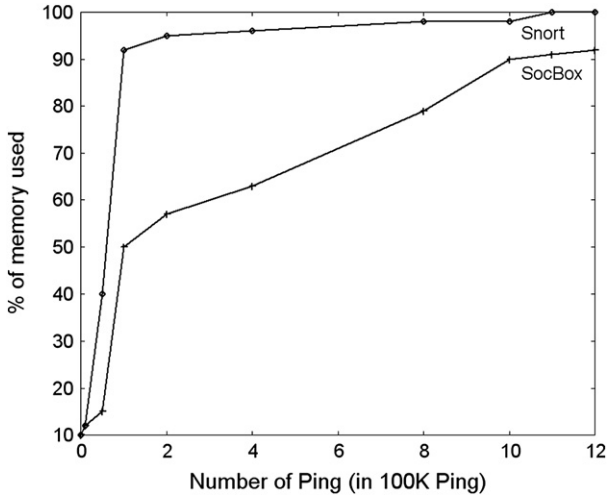

Fig. 4 – The SOCBox and Snort memory usage.

2.5. *Why new architecture?*

To overcome the limitations of the centralized SOCs, we propose a new distributed architecture called DSOC. This architecture is designed to be scalable, to support wide networks and to be highly available.

3. The DSOC architecture

3.1. *General view*

The DSOC is composed of four components based on the CIDF specifications (Staniford-Chen et al., 1998): data collectors (CBoxes), remote data collectors (R-CBoxes), Local Analyzers (LAs) and a Global Analyzer (GA). These global and simplified types of architecture are shown in Figs. 9 and 10.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

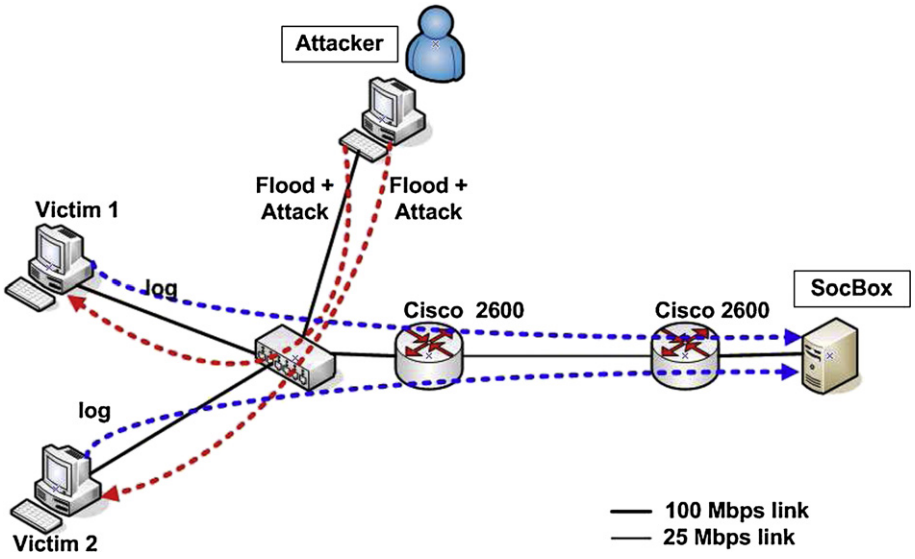

Fig. 5 – Remote SOCBox with syslog.

3.1.1. *Data collection box*

A CBox collects logs from sensors located on the same segment of a network. A sensor can be a host, a server, a firewall, an IDS or any system that generates logs. The advantage of our log collection approach is that no software has to be installed on the sensors. Moreover, our system is compatible with a wide range of hardware and software (Iv2 Technologies). A CBox formats logs and sends them to a local intrusion database (lidb). On each site we have one or several CBoxes and one of them acts as a Master CBox (M-CBox). The M-CBox is responsible for the management of all the CBoxes located on the same site. It polls regularly the other CBoxes and when a CBox is down, the M-CBox will collect data on the segment of the failed CBox. Each Master CBox also has a backup which polls it regularly and will become Master if need be.

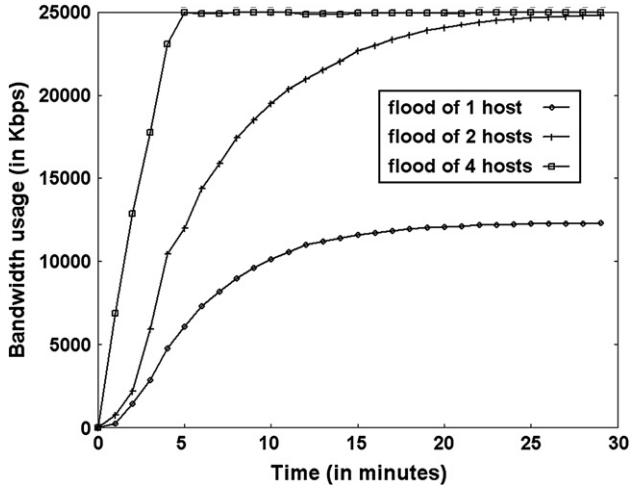

Fig. 6 – Bandwidth usage when flooding 1, 2 and 4 hosts.

3.1.2. *Remote data collector*

An R-CBox is a special CBox which collects logs coming from some critical sensors and from sensors hosting security tools on any site. Afterwards, logs are forwarded to the local intrusion database of another site and are analyzed to give the approximate security level of the concerned site in real-time. This helps to anticipate a reaction when a critical intrusion occurs or to investigate and troubleshoot a site that could be compromised, even if a hacker erases the logs on the compromised sensors (including the security tools).

3.1.3. *The local analyzer*

A Local Analyzer (LA) is responsible for intrusion detection on any site of a network. It analyzes formatted logs located in a local intrusion database (lidb) and generates alerts. Afterwards, it correlates the alerts to find more complex intrusions (intrusions composed of several events, distributed intrusions, intrusions directed to many sensors, etc.). The LA also compacts alerts by merging similar ones. All the alerts generated by an LA are sent to the global intrusion database (gidb). The gidb can have a mirror of itself for high availability purpose. The internal architecture of a Local Analyzer is described in Fig. 11.

3.1.4. *The Global Analyzer*

The Global Analyzer (GA) is a chosen LA responsible for global intrusion detection in a network. It analyzes alerts from the gidb, correlates and merges them if possible to generate optimized outputs. It is also able to detect more sophisticated intrusions that are directed to several sites. The GA regularly polls the other LAs and when one of them is down, the GA detects the occurring intrusion into the concerned site.

Another LA acts as the backup of the GA and polls it regularly. When the GA is down, the backup becomes the GA and another backup is elected. The DSOC architecture was designed bearing in mind that the data flow processed on the different sites of the network is not always homogeneous.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

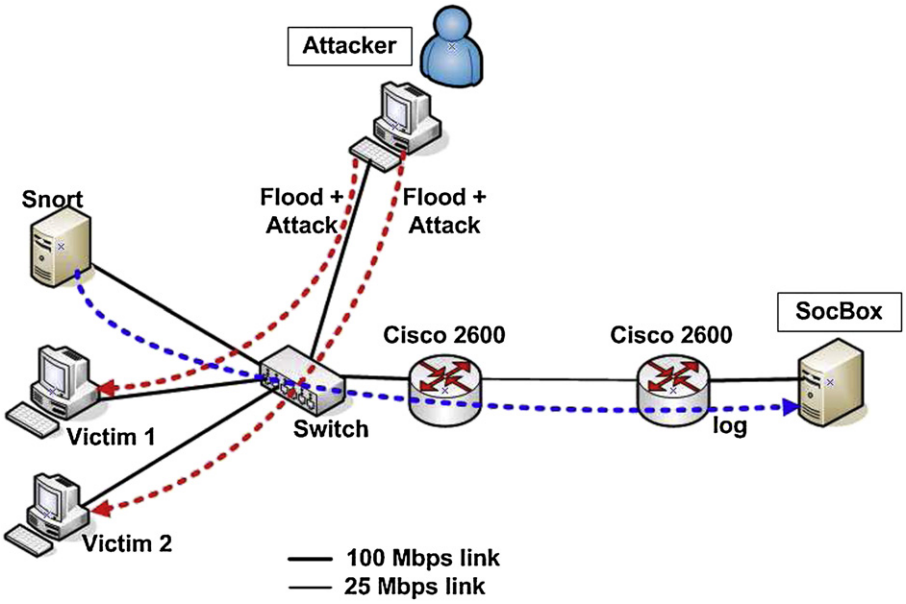

Fig. 7 – Remote SOCBox with Snort.

Indeed, on some sites, a large amount of data is processed and in this kind of situation, several CBoxes are needed for data gathering.

Even though a single CBox has to be installed on each segment, installing several CBoxes on the same segment is not excluded when the sensors located in this part of the network are operating under high workloads.

In quieter sites, only one CBox can be used to collect data coming from all the sensors.

The DSOC also implements the different types of boxes defined for network intrusion detection systems in (Northcutt and Novak, 2002). However, beside the pure technical aspects involved in such implementations, it is necessary to consider the supervision of an IT infrastructure as a full operational project.

3.2. *Data collection*

With DSOC, data is collected from heterogeneous sources using transport protocols such as syslog, snmp, smtp, html, etc. Data collection process is setups using two kinds of agents: protocol and application agents. The former collect information from sensors, the latter parse information for storage in a ''pseudo-standard'' format. These two modules are connected by a dispatcher. Such architecture allows high availability and load-balancing systems to be set at any level of the architecture.

3.2.1. *Protocol agents*

Protocol agents are designed to receive information from specific transport protocols (syslog, snmp, etc.). They act as server

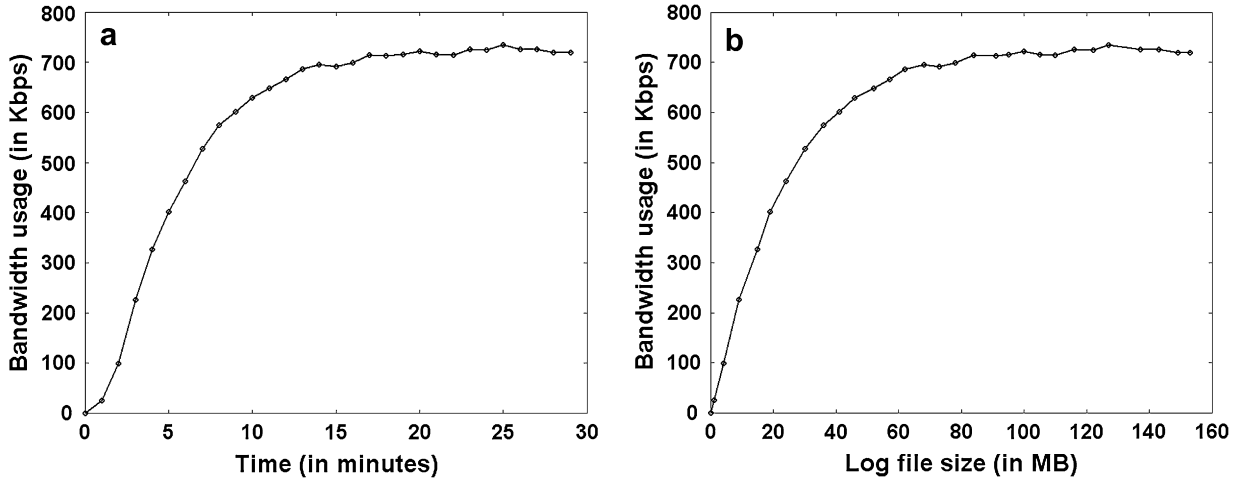

Fig. 8 – Bandwidth usage and log file size when Snort forwards data to a remote SOCBox. (a) Bandwidth usage when Snort forwards data to a remote SOCBox. (b) Log file size when Snort is used to forward data to a remote SOCBox.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

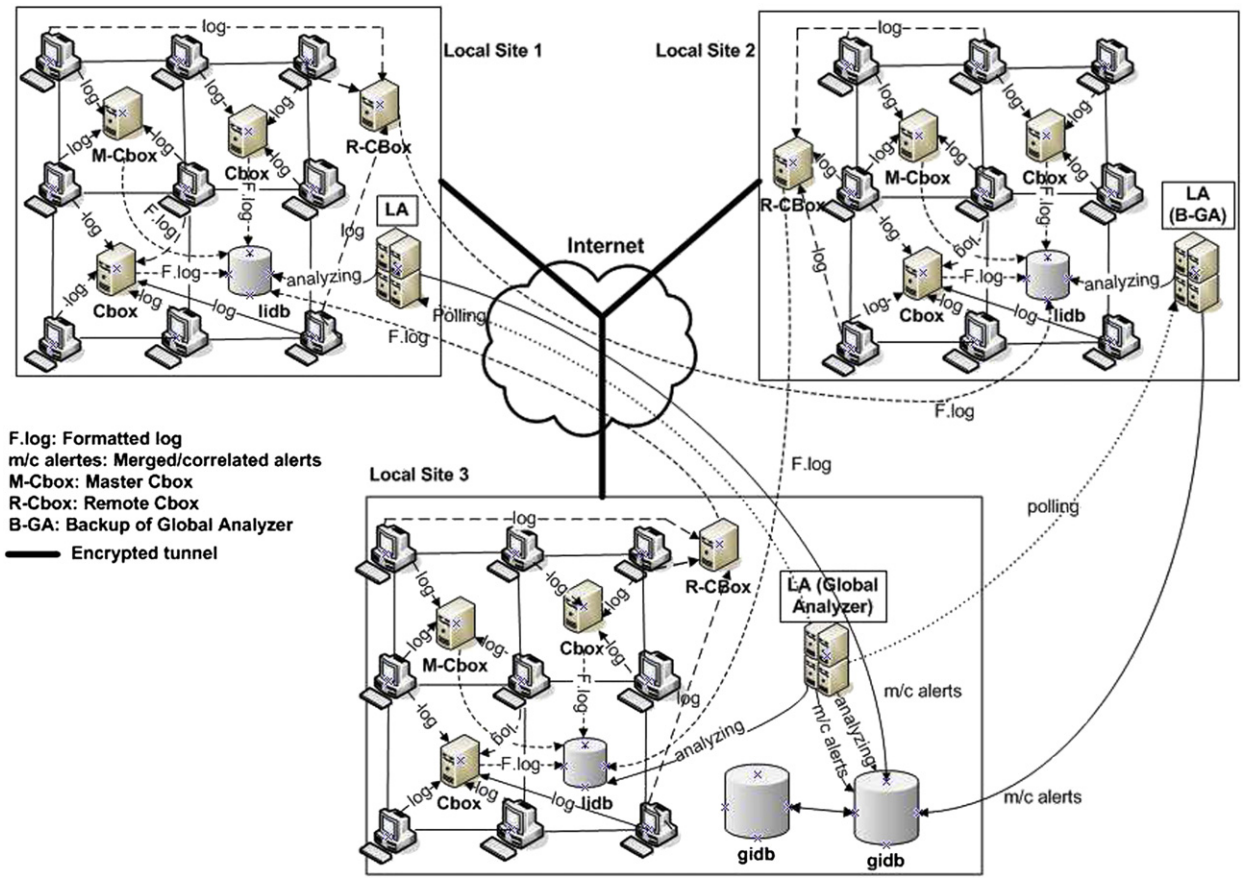

Fig. 9 – Global architecture of the DSOC.

side applications and their only purpose is to listen to incoming connections from sensors and make collected data available to the dispatcher. The simplicity of such agents makes them easy to implement and maintain.

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

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

3.2.2. *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. Here is the list of the autonomous operations performed by the dispatcher:

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

Application agents are formatting the messages so that they match 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 a local intrusion database.

3.2.3. *Messages*

Working with data generated by different types of equipments and transmitted via different transport protocols requires a ''standard'' formatting. Although an effort has been made to define a worldwide standard with IDMEF (Curry and Debar, 2003), it appears that the XML bus used is too heavy and resources consuming, principally for event correlation. However, a separate translation process must be implemented for IDMEF compliance. The DSOC database structure is given in Fig. 12.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

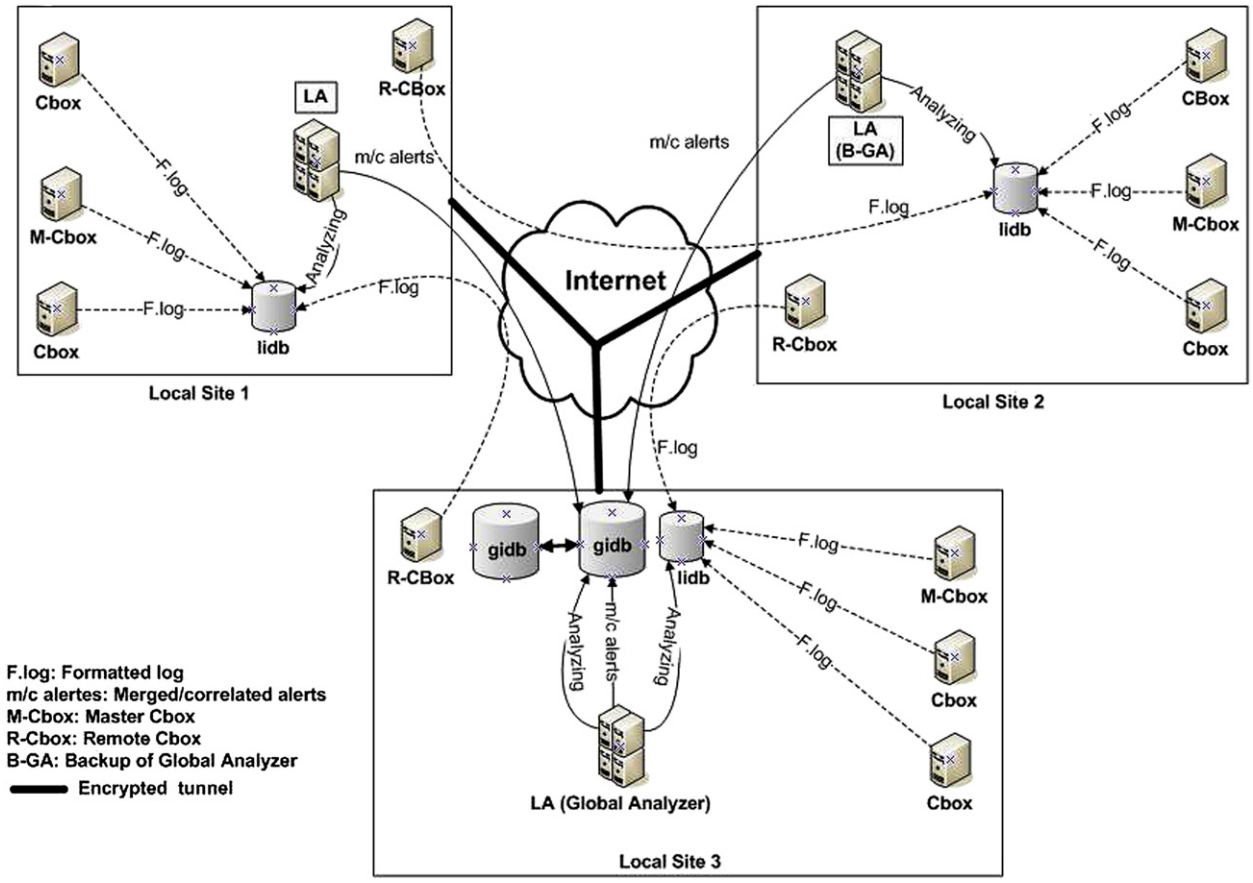

Fig. 10 – A simplified view of the DSOC.

3.2.4. *Example of a data collection on a Linux 2.6 system hosting an Apache 2.0 server*

Let us assume that a DSOC monitors the security of Linux 2.6 system hosting an Apache 2.0 web server. When a user executes an attack against the web server (a target identification for example), events about the attack are forwarded by syslog (acting as a transport agent) to a protocol agent of the DSOC data collection module. When this agent receives messages, it forwards them to the dispatcher which verifies their source and sees that they come from an Apache 2.0 server. After that, the dispatcher forwards messages to an Apache 2.0 application agent which parses them and puts them in the DSOC standard message format. This format is specially important for correlation operations.

All security messages coming from the Linux 2.6 system will be forwarded by the dispatcher to a Linux 2.6 application agent. This agent will be responsible for putting messages in the DSOC format.

Data collection process from a Linux 2.6 system hosting an Apache 2.0 server is showed on Fig. 13. To be analyzed, a message must be in DSOC standard format.

3.2.5. *Example of a Snort 1.8 alert formatting*

The following lines show an example of a Snort 1.8 alert in syslog format:

```
10.1.62.90:<33>snort[29036]: [1:974:2] WEB-IIS ... access
[Classification: Attempted Information Leak] [Priority: 3]: {TCP}
10.1.21.186:4597 → 10.1.62.90:80
```

Based on regular expression in Perl, the following operations are performed by a DSOC dispatcher to identify a Snort 1.8.x alert in syslog format.

```
if ($line =~/.*snort: \[\d+:\d+:\d+\] .*){
   send_to_snort_1.8_application_agent($line)}
}
```

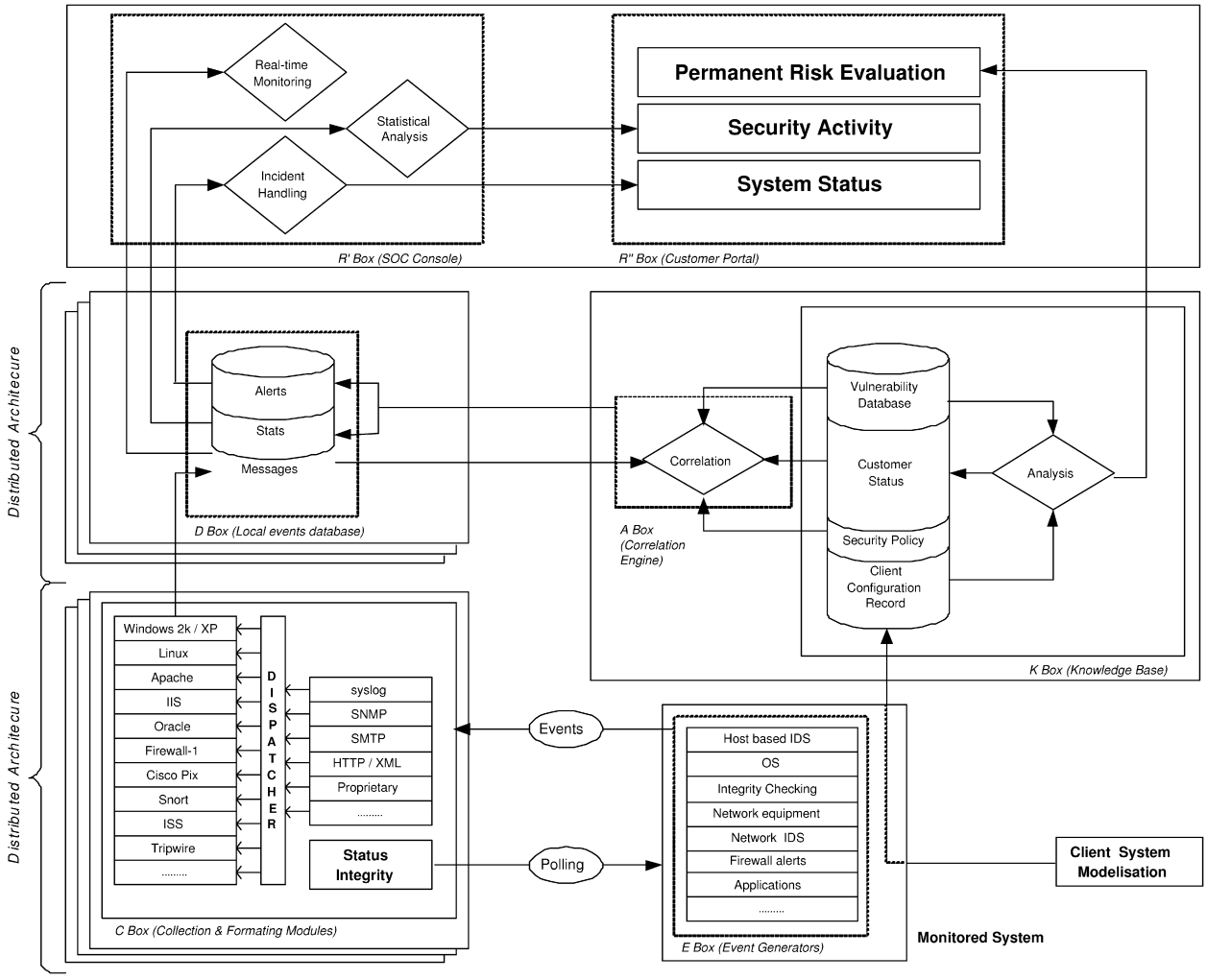

Fig. 11 – The internal architecture of a local analyzer.

The application agent *send_to_snort_1.8_application_agent* would perform the following operation to put the Snort 1.8 alert in the DSOC standard message format:

```
MSG_TYPE = ''Snort 1.8'';
if($line =~/(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}):.*snort.*
\[\d+: (\d+):\d+\] (.*) \ [Classification: (.*)\] \ [Priority:. *\]:?
\{(.*)\} (.*):(.*) → (.*):(.*)/) {
   $sensor_id = $1;
   $msg_type = $MSG_TYPE;
   $epoch_time = timelocal((localtime) [0,1,2,3,4,5]);
   $source = $6;
   $target = $8; $proto = $5; $src_port = $7; $tgt_port = $9;
   $intrusion_type = $intrusion_type [SnortIntrusionType($2)];
}
```

3.3. *Correlation overview*

The correlation 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 operation is to identify duplicates, set a specific flag in order to keep the information and continue without keeping multiple identical messages.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

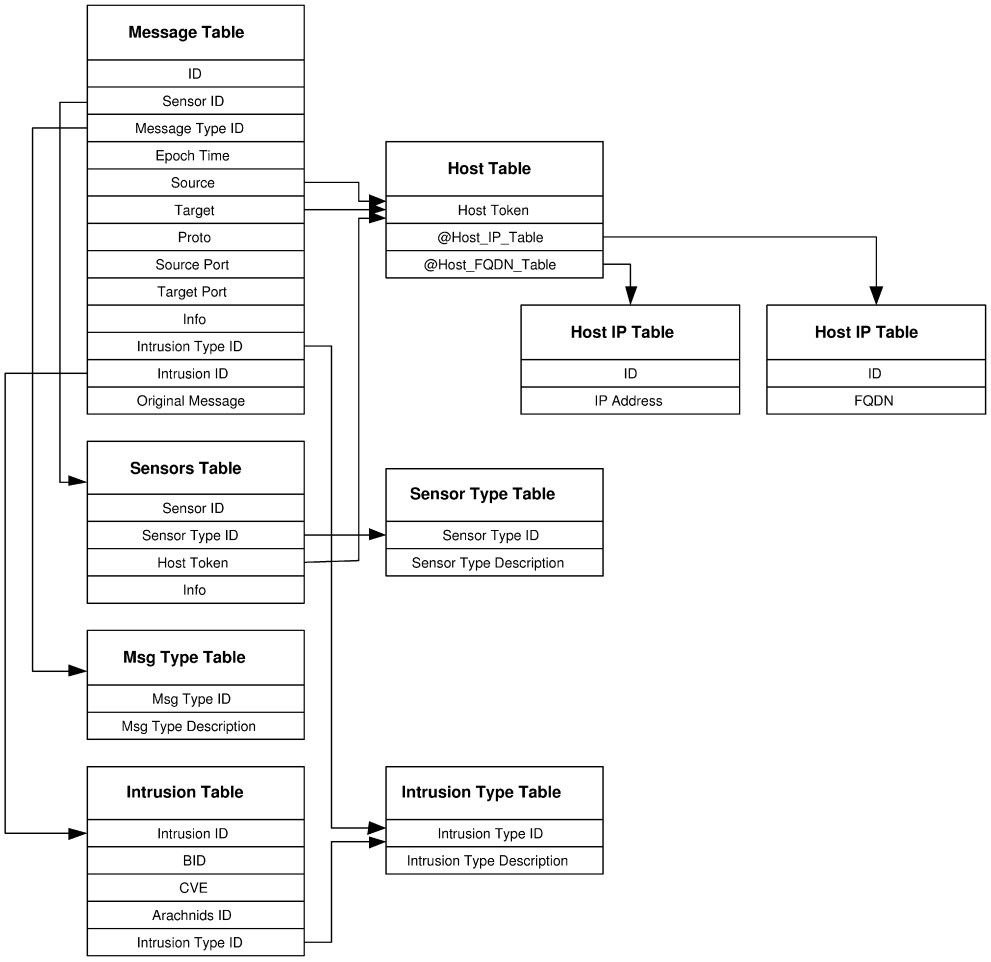

Fig. 12 – Formatted message definition structures.

- sequence pattern 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 ongoing 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 management, as well as slow and distributed intrusion processes.
- system exposure and criticality analysis provide information about the target system vulnerability to detect intrusion attempts. Indeed, it seems in appropriate to have the DSOC generating alerts concerning an intrusion scenario based on 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.

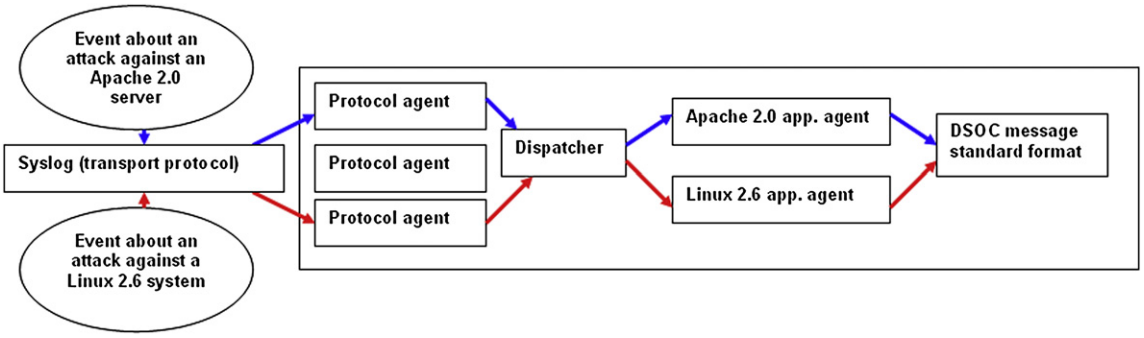

Fig. 13 – Data collection from on a Linux 2.6 system hosting an Apache 2.0 server.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

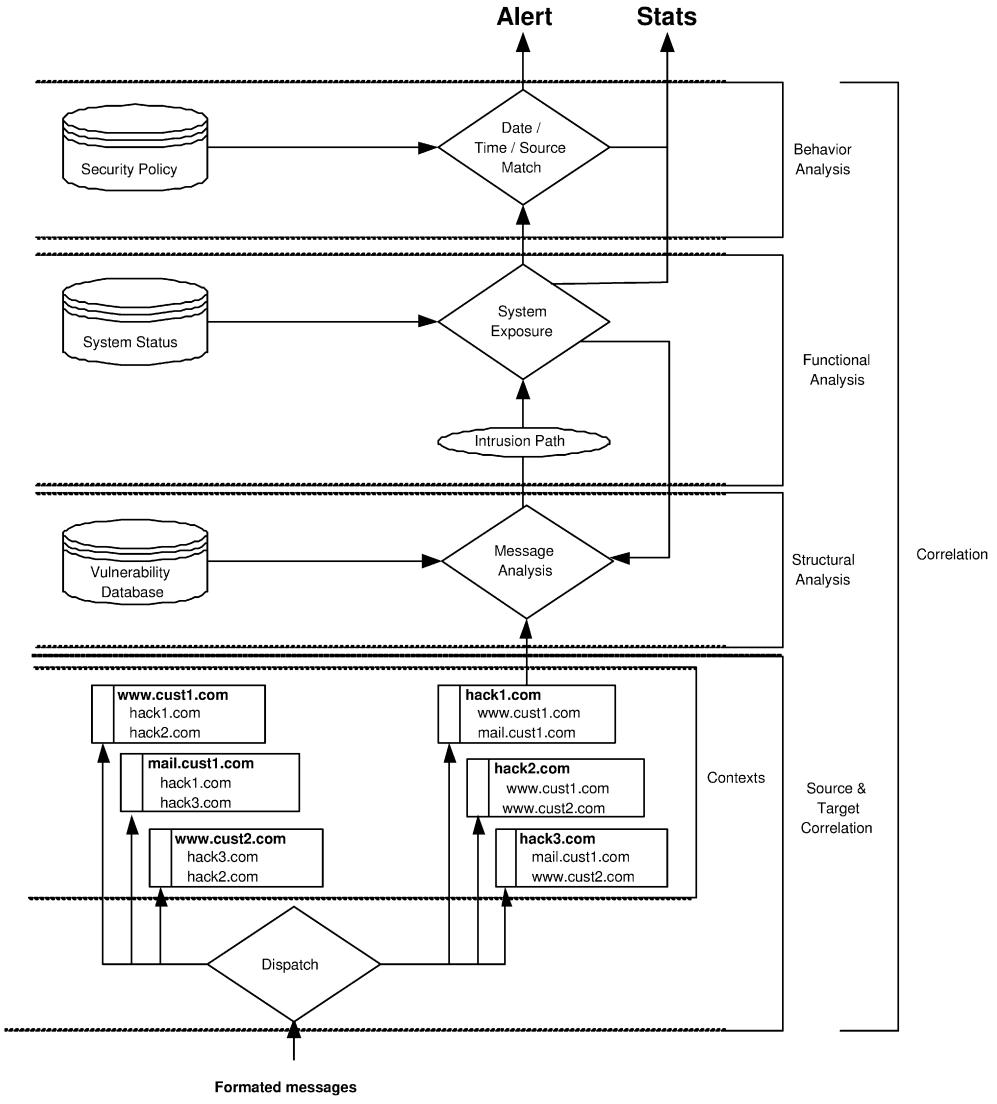

Fig. 14 – Main correlation operations.

- 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 Fig. 14.

3.3.1. *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 (same source, same target, same time, etc.). 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. Another important characteristic of a 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 ongoing 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.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

- 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 Fig. 15.

3.3.2. *Example of a time-base context*

For example, events about multiple DOS attacks against a same host and occurring approximately in the same time will causes the creation of a time-based context. A correlation using this context will generates a unique DDOS alert.

3.4. *Data analysis*

3.4.1. *Structural and behavior-lead alerts*

To generate alerts, the operations performed are the following: correlation, structural analysis, intrusion path analysis and behavior analysis. Correlation is a stand-alone operation leading to the creation of contexts in 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 an attack tree (Schneier, 1999). Intrusion path analysis is the next step the output of which provides the detected intrusion attempt with information about the exposure of the targeted system. 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 do 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 the criticality of target systems.

3.4.2. *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 in each context. Each module is activated by a specific message

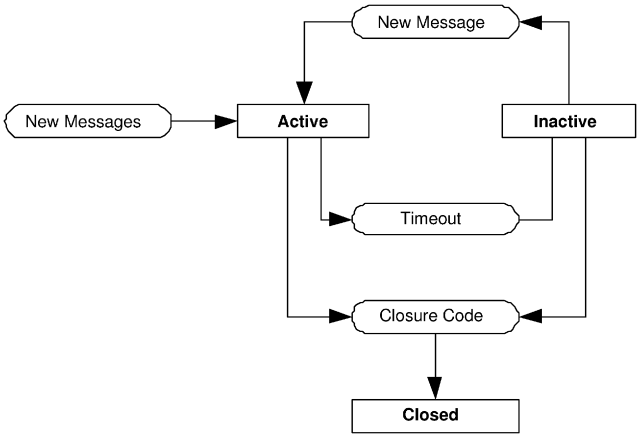

Fig. 15 – Context status management.

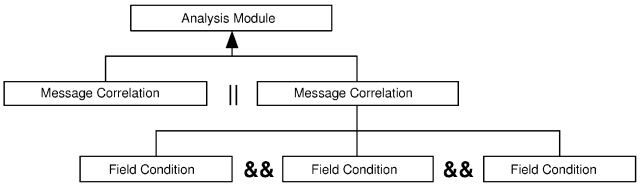

Fig. 16 – Analysis module structure.

and performs analysis using ''standard'' semantics. The output of the analysis modules is the result of several logical operations between autonomous conditions against fields of contexts. Fig. 16 describes members of such operations. 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 performance 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 the 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 intrusion detection such as (slow) portscans, brute forcing, etc.

3.4.3. *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.

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.

3.4.4. *Example of correlation: detection of a distributed attack*

The evaluation of the correlation capability of our system is described on Fig. 17. The test take place in a real ISP network which manages more than 50,000 subscribers. That ISP network is composed by a core sub-net and several regional sub-nets.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

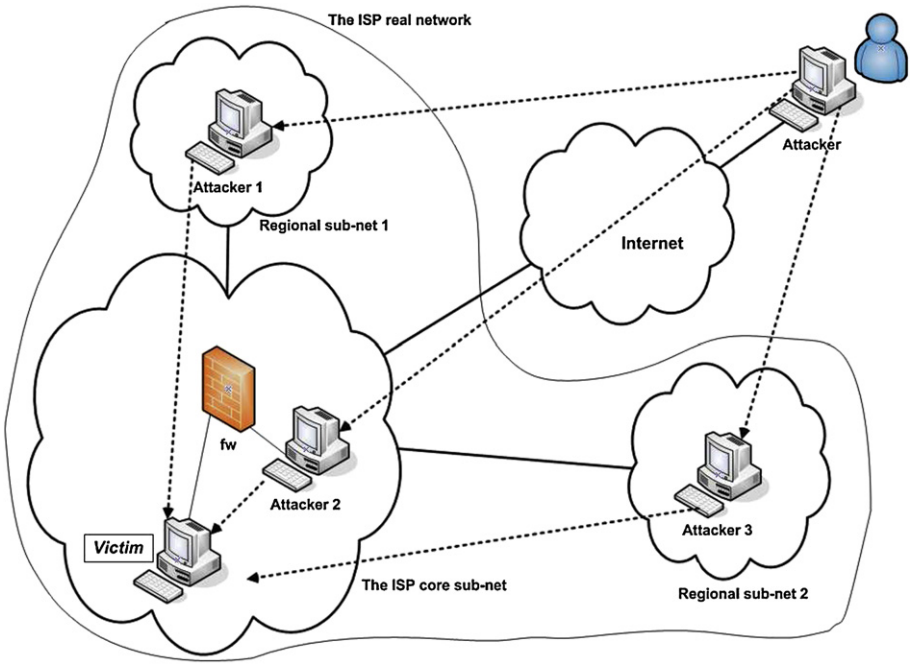

Fig. 17 – Evaluation of the correlation capability of the DSOC.

The scenario of this attack is the following:

From an external host, a user called *Attacker* compromises three less secured hosts on the ISP network (*Attacker 1* in regional sub-net 1, *Attacker 2* in the core sub-net and *Attacker 3* in regional sub-net 2). From these hosts, he launches an attack against a host named *Victim* and located in the core sub-net. The attack is composed of the following actions:

- From *Attacker 1* located in regional sub-net 1, he executes a scan (with nmap) to detect open ports on *Victim*. He sees that http and ssh are open on *Victim*.
- From *Attacker 2* located in the core sub-net, he executes a Nikto Attack against *Victim*.
- From *Attacker 3* located in regional sub-net 2, he tries to gain access to *Victim* using ssh.

In this test, the firewall (Cisco Pix) detects the scan and a Snort sensor located in the core sub-net generates an alert about the Nikto Attack. The alerts are sent to the DSOC. The attempt to gain access to *Victim* is logged by syslog and this information is forwarded to the DSOC

Because these three actions have the same target (*Victim*) and they take place approximatively in the same time, our system creates a unique context which includes them and performs a time-based correlation followed by a target based correlation. Then, it generates a ''suspicious behavior'' alert about actions coming from Attacker and targeted to *Victim*. An alarm is also sent to the security manager for advanced investigation on *Attacker 1*, *Attacker 2* and *Attacker 3*.

Without correlation, it would be impossible to detect this attack. Our system is thus able to correlate alerts coming from divers sources (firewalls, IDS, hosts, etc.) to generate a single alert. Many NIDS cannot detect this kind of multi-events intrusion.

3.5. *Security assessment*

Ensuring that all goes well on any site is essential to the monitoring of the security activity in a multi-site network. The R-CBoxes are built for this purpose. While a CBox sends data to the local intrusion database (lidb) of the site where it is located, an R-CBox forwards data to the lidb of a remote site. The analysis of forwarded data gives an approximate view of the security level of the concerned site. When an incident occurs, this can help with troubleshooting. Operations related to a site security level assessment are the following:

- In each site S, the R-CBox collects data from the LA and some critical sensors and sends it to the lidb of another site.
- The LA of the remote site which receives data from the R-CBox analyzes it and generates alerts (each alert has a level of criticality). Afterwards, an approximate security level of site S is determined.
- The LA of site S analyzes data gathered by CBoxes, finds intrusion patterns on it and detects suspicious behavior. It also determines the real security level of the site.
- The GA compares these two security levels and when there is a significant deviation between them, a suspicious behavior alert is generated. A significant deviation can be a sign of the R-CBox or the LA compromise. This can also be due to the fact that an intruder attempts to hide the compromise of one or several sensors. In this case, an alarm is sent to the security manager for advanced investigations.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

3.5.1. *The site security level assessment process.*

The site security level assessment process aims at making sure that everything is working well in every branch of a network.

In order to achieve this, we determine an approximate level of security for each site according to the data gathered by the R-CBoxes and an accurate level of security based on the alerts generated by the CBoxes. We then compare the two values. Given that the R-CBoxes are collecting their data from the most critical sensors of the sites (the ones that are the most likely to attract hackers), the estimated level of security should be approximately equal to the official level of security. If there is a big abnormality between the two types of security level, then there is a problem on the concerned site; should it be the case, an alarm is generated in order to carry out an in-depth inspection of the vulnerable site.

3.5.2. *Evaluation of the real level of a site security*

Let *X*, *Y* and *Z* be a network site, a sensor, an alert respectively and:

Xsl: the theoretical security level of a network site.

Ylsl: the theoretical local security level of a sensor (OS point of view). We suppose that the more important a sensor is, the more secure it is.

Yial: the theoretical capacity of a sensor to access the hosts of the site where it is located.

Yeal: the theoretical capacity of a sensor to access the hosts of the network other sites.

Zcl: the theoretical criticality level of each alert.

L, *M*, *H*: respectively, low, medium and high (the security level of each alert).

- The theoretical security level (Ytsl) of a sensor is given by $\mathrm{Ytsl} = \mathrm{Xsl} \times \mathrm{Ylsl}$ with:

$$\mathrm{Ytsl} = \begin{cases} L & \text{if } \mathrm{Xsl} = L \ \&\ \mathrm{Ylsl} = L. \\ H & \begin{cases} \text{if } \mathrm{Xsl} = \begin{vmatrix} H \\ M \end{vmatrix} \ \&\ \mathrm{Ylsl} = H \parallel \\ \text{if } \mathrm{Xsl} = H \ \&\ \mathrm{Ylsl} = M. \end{cases} \\ M & \text{in the other cases.} \end{cases}$$

- The theoretical capacity of a sensor to access all the hosts of a network (Ytac) is given by $\mathrm{Ytac} = \mathrm{Yial} \times \mathrm{Yeal}$ with:

$$\mathrm{Ytac} = \begin{cases} L & \text{if } \mathrm{Yial} = L \ \&\ \mathrm{Yeal} = L. \\ H & \begin{cases} \text{if } \mathrm{Yial} = \begin{matrix} H \\ M \end{matrix} \ \&\ \mathrm{Yeal} = H \parallel \\ \text{if } \mathrm{Yial} = H \ \&\ \mathrm{Yeal} = M. \end{cases} \\ M & \text{in the other cases.} \end{cases}$$

- The real criticality level (Zrlc) of an alert *Z* on a sensor *Y* is given by $\mathrm{Zrlc} = \mathrm{Ytsl} \times \mathrm{Ytac} \times \mathrm{Zlc}$ with:

$$\mathrm{Zrlc} = \begin{cases} L \text{ if } \mathrm{Zlc} = L \ \forall \mathrm{Ytsl} \text{ and } \mathrm{Ytac}. \\ M \begin{cases} \text{if } \mathrm{Zlc} = M \ \forall \mathrm{Ytsl} \text{ and } \mathrm{Ytac} \parallel \\ \text{if } \mathrm{Zlc} = H \ \&\ \mathrm{Ytsl} = H \ \&\ \mathrm{Ytac} = \begin{vmatrix} M \\ L \end{vmatrix}. \end{cases} \\ H \text{ if } \mathrm{Zlc} = H \ \&\ \mathrm{Ytsl} = \begin{vmatrix} L \\ M \end{vmatrix} \forall \mathrm{Ytac} \parallel \\ H \text{ if } \mathrm{Zlc} = H \ \&\ \mathrm{Ytsl} = H \ \&\ \mathrm{Ytac} = H. \end{cases}$$

- The real level of security of a site *X* (Xrsl) is given by

$$\mathrm{Xrsl} = \begin{cases} H \text{ if } \mathtt{T}(X, (\mathrm{Zrlc}_i)_{1 \le i \le n}) = L. \\ M \text{ if } \mathtt{T}(X, (\mathrm{Zrlc}_i)_{1 \le i \le n}) = M. \\ L \text{ if } \mathtt{T}(X, (\mathrm{Zrlc}_i)_{1 \le i \le n}) = H. \end{cases}$$

with $\mathtt{T}(\mathtt{A}, \mathtt{B})$ a function which gives the greatest criticality level of all the generated alerts (B) for a site A. This function is executed by the LAs.

3.5.3. *Dimensioning the R-CBoxes*

In order to analyze the data collected by the R-CBoxes thus enabling us to give an approximate view of the security level of a site X (called Xasl), the R-CBoxes must collect data from sensors which are the most likely to attract hackers (sensors hosting confidential or important data) and sensors hosting security tools (LAs, Firewalls, IDS, etc.). The approximate security level (Xasl) of a site *X* is given by

$$\mathrm{Xasl} = \begin{cases} H \text{ if } \mathtt{T}(X, (\mathrm{Zrlc}_i)_{1 \le i \le n}) = L. \\ M \text{ if } \mathtt{T}(X, (\mathrm{Zrlc}_i)_{1 \le i \le n}) = M. \\ L \text{ if } \mathtt{T}(X, (\mathrm{Zrlc}_i)_{1 \le i \le n}) = H. \end{cases}$$

with $\mathtt{T}(\mathtt{A}, \mathtt{B})$ a function which gives the greatest criticality level of all the alerts (B) generated by analyzing the logs gathered by the R-CBoxes of a remote site A. Under normal conditions, we have

$$\mathrm{Xasl} \approx \mathrm{Xrsl}.$$

A great deviation between Xrsl and Xasl generates an alarm; this can be the sign of one or several R-CBoxes being compromised. This can also be an attempt to hide the compromise of one or several sensors.

3.5.4. *Protecting the communications between the DSOC components*

One of the key points here is to make sure that no illegitimate computer will act as an LA, a CBox or an R-CBox in order to get privileged access to the system. To ensure the security of our system, any LA or any R-CBox that needs to exchange information with another LA has to use a certificate to prove its identity. Moreover, all the communications between the LAs (also including the GA) and the communications between the R-CBoxes and the LAs will have to pass through an encrypted tunnel, available via the SSL protocol.

4. Experiment results

4.1. *Detection of an intrusion which uses a relay*

The intrusion detection capabilities and the performance of the Local Analyzers were studied in our recent work (Ganame et al., 2006). The goal of the present evaluation is to check the aptitude of the DSOC for detecting an intrusion which uses a relay (Fig. 18).

The scenario of this attack is the following:

Attacker wants to hack a host (called *Victim*) located on the ISP site and hosting information about subscribers. His idea is to gain access to *Victim* by brute force attack and to steal data about subscribers. The ISP network is well secured and *Victim* can be accessed only from special hosts on the Management

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

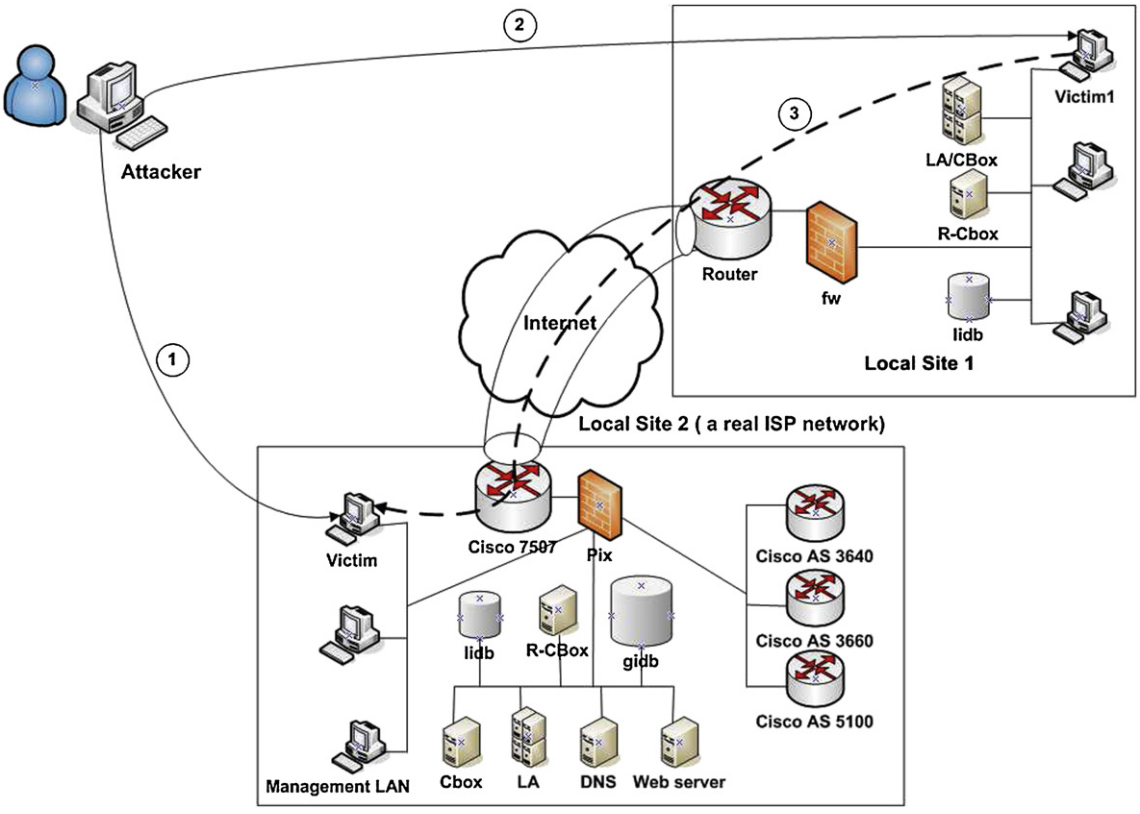

Fig. 18 – Evaluation of the DSOC aptitude for detecting an intrusion which uses a relay.

LAN (for maintenance purpose) and on our lab site. *Attacker* tries to compromise *Victim* and unfortunately for him, all his actions are refused. On second thoughts, he infers that it would be easier for him to try to hack *Victim* from hosts located on another network site. After multiple attempts, he compromises a host (*Victim1*) on our lab site (less secured for the evaluation purpose). Afterwards, he launches the attack consisting of the following actions:

- From *Victim1*, he executes a quick scan with nmap to detect open ports on *Victim*. He sees that ssh and mysql are open on *Victim*.
- From *Victim1* he launches a brute force attack with THC-Hydra (THC, 2006) to gain access to the mysql database of *Victim*.

During this test, the behavior of the DSOC is the following:

(1) The ISP site LA generates alerts about the brute force attack on *Victim* (multiple ''authentication failed'' messages are sent to the local intrusion database and a unique alert ''brute force attack'' is sent to the global intrusion database).

(2)
- The LA of our lab site generates alerts about the multiple attempts at access to *Victim1* (''portscan detection'', some ''authentication failed'', a ''authentication successful'' and ''su: privileges gained''). These alerts are sent to the global intrusion database.
- The R-CBox of our lab site gathers data from the LA, the Firewall and *Victim1* and sends it to the local intrusion database of the ISP site. Data is about actions like ''portscan'' and ''authentication failed''.
- The ISP site LA generates alerts by analyzing data gathered by our lab site R-CBox and gives an approximate security level of our lab site.

(3) The ISP site LA generates alerts about the scan (detected by the firewall) and the brute force attack (detected by analyzing *Victim* logs). These alerts are sent to the global intrusion database.

Comments. Because the scan and the brute force attack have the same target and they take place approximately at the same time, the LA of the ISP site matches them with the same context (time-based correlation) and generates a unique alert.

Due to the fact that one of the steps of the attack (the brute force attack) is carried out by 2 hosts (*Attacker* and *Victim1*) on the same target (*Victim*), a second correlation is performed by the ISP site LA (acting as the GA). The purpose of this correlation is to verify if there exists a relation between both attempts at brute force attack. The GA sees that *Attacker* attempted to access *Victim1*. Then, it generates a ''suspicious behavior'' alert about a ''probable compromission of *Victim 1*''. An information alarm is also sent to the security manager for advanced investigation of *Victim1*.

In short, we can say that our system is able to detect an attack which occurs simultaneously on several sites of a network. Most IDs cannot detect this kind of attacks.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

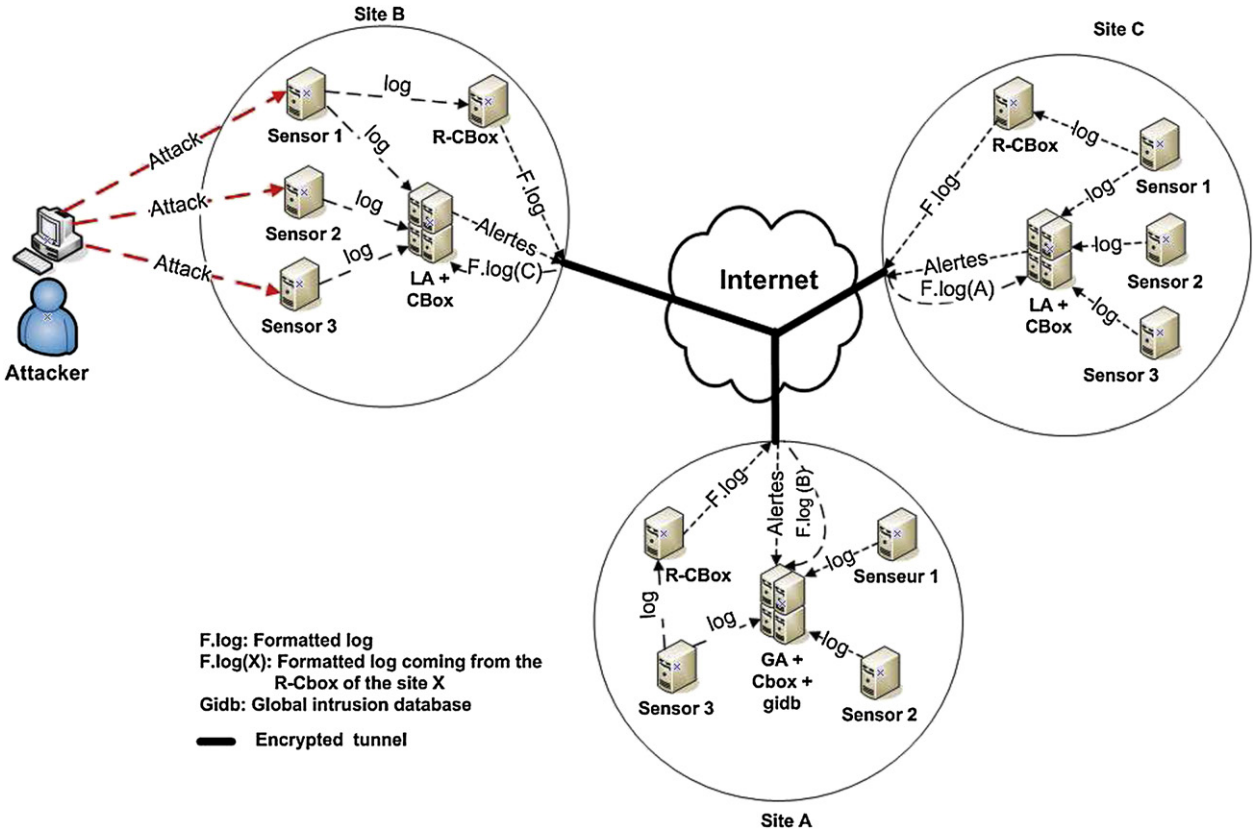

Fig. 19 – Isolation attack of a site with the DSOC.

4.2. *Behavior of the DSOC when a strong attack occurs on a site*

To verify the DSOC behavior when a strong attack occurs on a network site, the following test is carried out.

A hacker initiates an attack on a network composed of 3 sites connected by VPN links (Fig. 19).

The Global Analyzer (GA) is installed on site A, a local analyzer (LA) is installed on site B, another LA is installed on site C and Scanlogd (Openwall-Project, 2006) is installed on the sensors to detect the portscans.

After scanning the network, the hacker sees open ports on sensors located on site B and decides to hack this site. Using a traffic generator, he floods site B. The goal of this operation is to camouflage his intrusion in a high data flow. Then, he launches an attack on site B to steal data.

The behavior of the DSOC. During the scan of site B, the LA of this site detects the scan (data about the scan is collected by Scanlogd) and generates alerts. The R-CBox receives data from some critical sensors and forwards it to the LA of site A (acting as the GA). The GA analyzes data coming from the R-CBox of the site B and detects that a scan is occurring on this site. After that, it evaluates the approximate level of site B security and concludes that there is no significant intrusion activity on site B. Operations performed on each site during this attack are the following:

4.2.1. *On site B*

- The LA generates the alerts, merges and correlates them. Then, it assesses the real level of security of site B and forwards this information to the GA. Afterwards, it forwards the alerts to the global intrusion database (gidb) located on site A. Only merged and correlated alerts are transmitted to the gidb via the VPN link. This minimizes the communication through the VPN and optimizes the link use.
- The R-CBox of the site B gathers data coming from some critical sensors and sends them to GA.

4.2.2. *On site A*

- The GA analyzes data coming from the site B R-CBox and generates alerts. Then, it determines site B approximate security level. It sees that intrusive activity is occurring on site B. For this reason, the GA sends an information alarm to the security administrator. Afterwards, it compares site B real security level to the approximate one and sees that they are similar. The GA concludes that neither the LA nor the R-CBox is compromised.

The security administrator being informed by the GA that an intrusion is in progress on site B, he blacklists the source

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

of the attack and analyzes the alerts to see the actions carried out by the hacker.

Comments. The bandwidth usage during this test is shown in Fig. 20. The graph in this figure can be divided into 3 parts:

- In the first part, the alerts generated by site B LA about the portscan are forwarded to the GA located on a remote site. This transfer of data uses 9 kbps of bandwidth.
- In the second part of the graph, the site B R-CBox gathers data about the portscan and sends it to the GA. This transfer of data uses 14 kbps of bandwidth.
- The third part of the graph presents the bandwidth usage when site B LA transfers data about the attacks carried out on the sensors to the Global Analyzer (GA).

We notice a variation of the bandwidth usage during the test with peaks around 55 kbps when both the LA and the R-CBox forward data to the GA.

This test also shows that the DSOC makes it possible to see in real-time that an intrusion is in progress on site B. This permits to blacklist the source of the attack and to prevent the theft of data. A centralized SOC is unable to detect this kind of attack.

4.3. *Comparison between the SOCBox and the DSOC bandwidth usage*

The comparison between the SOCBox and the DSOC bandwidth usage is shown in Fig. 21.

The first curve (at the top of Fig. 21) shows the bandwidth usage when 2 sensors located on the same site send their logs to a centralized SOC located on another site (for more details, see Fig. 1). The results are obtained by flooding the sensors with Ping packets of 1460 bytes for 30 min. The sites are connected by a 25 Mbps link.

The second curve (at the bottom of Fig. 21) shows the bandwidth usage when a DSOC monitors the security of 2 sites A and B (more details about the test are given in Fig. 19). The GA is located on site A and an LA is installed on site B. The sensors are flooded in the same conditions as the centralized SOC (first curve) and some attacks are carried out on the sensors during the flood.

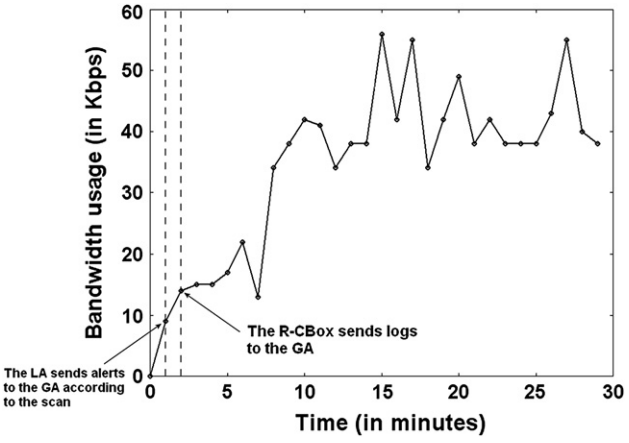

Fig. 20 – Bandwidth usage when an LA forwards alerts to the GA.

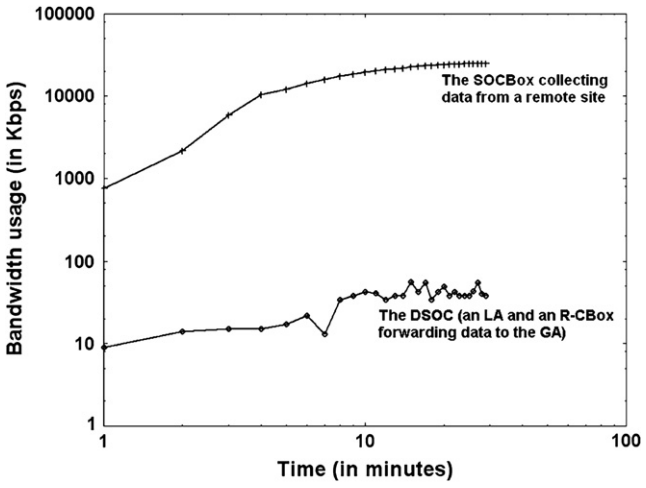

Fig. 21 – Comparison between the SOCBox and the DSOC bandwidth usage.

In this test, any centralized SOC would give the same result as the SOCBox.

Conclusion. With the DSOC, we use around 443 times less bandwidth than with a centralized SOC. This is explained by the following facts:

- When a centralized SOC gathers data from a remote site, data is forwarded to it in a raw format. The stronger an attack, the greater amount of data is forwarded.
- With the DSOC, intrusions are detected by the LA locally on site B. Afterwards, the alerts are merged and correlated before being transmitted to the GA. This reduces the quantity of data to be forwarded to the GA.

5. Related work

To overcome the limitations of the traditional IDS which are unable to detect complex attacks, several types of IDS have been proposed and tested like distributed ones (Ajith et al., 2005; Lee et al., 2005; Li et al., 2004). DSCIDS (Ajith et al., 2005) is a distributed IDS composed of two elements: intelligent agents which collect data on each host and send them to a central unit called Analyzer/Controller. The Analyzers/Controllers are built hierarchically and each one manages several collection agents. They are also responsible for data analysis and alert correlation. To detect complex intrusions, the collaboration of several intelligent agents is needed. Lee et al. (2005) proposed a distributed intrusion detection system which correlates alerts in real-time. In each network, a sensor collects data and eliminates redundant information. Afterwards data is analyzed for intrusion detection. Correlation is carried out to detect complex intrusions like distributed ones and if there is great similarity between several alerts, they are merged.

Other methods based on a P2P approach were proposed for scalability purpose. One of the best-known methods is INDRA (Janakiraman et al., 2003). With INDRA, each host runs a daemon which analyzes local intrusions and provides controlled access to resources. When a host detects an intrusion,

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004

a multi-cast message is sent to the other hosts which check the integrity of the message and blacklist the source of the intrusion if need be.

Cooperation of IDSs is still ongoing work (Gowadia et al., 2005) (Yu et al., 2005). PAID (Gowadia et al., 2005) is a cooperative agent-based intrusion detection system. In PAID architecture, each agent is autonomous. It detects intrusions and collaborates with the other agents to detect complex intrusions. With TRINETR (Yu et al., 2005), an intelligent agent collects data on each host of a network and sends it to a coordination agent which analyzes data to find intrusion patterns. When the coordination agent needs information to deduce whether or not there is an intrusion, it can request a particular collection agent.

6. Conclusion

Intrusions are clearly taking place and thus there is a need for operational supervision systems today. Our DSOC demonstrates its capability to detect intrusions and to present their status clearly. It also proves its ability to compact similar alerts and to correlate alerts coming from heterogenous platforms on several sites to detect more complex intrusions.

However, the development of some functionalities of the Global Analyzer (the management of the LAs) must be accomplished to make our system entirely operational. This will ensure the system scalability and messages will be processed better.

REFERENCES

Ajith A, Jain R, Johnson T, Sang YH, Sanyal S. D-SCIDS: distributed soft computing intrusion detection system. Journal of Network and Computer Applications 2005. Elsevier Science Direct.

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

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

Ganame AK, Bourgeois J, Bidou R, Spies F. Evaluation of the intrusion detection capabilities and performance of a security operation center. In: International Conference on Security and Cryptography. INSTICC Press; August 2006a. p. 48–55.

Ganame AK, Bourgeois J, Bidou R, Spies F. A high performance system for intrusion detection and reaction management. Journal of Information Assurance and Security Sep 2006b;3: 181–94.

Gowadia V, Farkas C, Valtorta M. PAID: a probabilistic agent-based intrusion detection system. Computers & Security 2005; 24(7):529–45.

Janakiraman R, Waldvogel M, Zhang Q. Indra: a peer-to-peer approach to network intrusion detection and prevention. In: Proceedings of IEEE WETICE; June 2003.

Lee S, Chung B, Kim H, Lee Y, Park C, Yoon H. Real-time analysis of intrusion detection alerts via correlation. Computers & Security Mai 2005.

Li C, Song Q, Zhang C. MA-IDS architecture for distributed intrusion detection using mobile agents. In: Proc. of the 2nd International Conference on Information Technology for Application (ICITA), Mai 2004. p. 451–55.

Northcutt Stephen, Novak Judy. Network intrusion detection. 3rd ed. New Riders; September 2002.

Openwall-project, Scanlogd 2.2.6: a port scan detection tool, <http://www.openwall.com/scanlogd>; 2006.

Puppy RF. Nikto 1.35: an open source web server scanner, Available from: <http://www.cirt.net/code/nikto.html>; 2006.

Schneier B. Attacks trees. Dr. Dobb's Journal 1999.

Staniford-Chen S, Tung B, Schnackenberg D. The common intrusion detection framework (cidf). In: Information Survivability Workshop, Orlando, October 1998.

Iv2 technologies, <http://www.iv2-technologies.com>.

THC. The hacker's choice, thc releases, thc-hydra v5.2, Available from: <http://www.thc.org/releases.php>; 2006.

Yu J, Reddy YV, Selliah S, Reddy S, Bharadwaj V, Kankanahalli S. TRINETR: an architecture for collaborative intrusion detection and knowledge-based alert evaluation. Advanced Engineering Informatics 2005;19(2):93–101.

Zti-Telecom. Ip traffic (2.3), a test and measure tool, Available from: <http://www.zti-telecom.com/fr/pages/iptraffic-test-measure.htm>; 2005.

Please cite this article in press as: Karim Ganame A et al., A global security architecture for intrusion detection on computer networks, Comput. Secur. (2008), doi:10.1016/j.cose.2008.03.004