

# Benchmarking a distributed intrusion detection system based on ASAX: Preliminary results

*Véronique Abily and Mireille Ducassé*

IRISA/INSA de Rennes, Campus Universitaire de Beaulieu,  
F - 35042 Rennes Cedex, France, email: {abily ducasse}@irisa.fr

April 29, 2000

## 1 Introduction

The objective of the experimentation described in this abstract is to assess the feasibility of a sophisticated distributed intrusion detection system (DIDS). Indeed, informal discussions with system engineers show that they are reluctant to set up a DIDS. As a matter of fact, designing and maintaining a DIDS on a real life scale is a non trivial exercise. Most people fear that it would not be worth the effort, because the DIDS would make the performances of the audited network collapse. We believe that it is essential to have rational data about the costs of a DIDS.

ASAX (Advanced Security audit trail Analysis on uniX) is a research prototype which can detect intrusion signatures in audit trails. The detection mechanism is based on rules programmed in a C-like dedicated language [2]. With these rules, IDS systems can be built which can detect complicated and intertwined intrusion scenarios, in particular scenarios which require several actions. Informal preliminary experiments hinted that ASAX could be reasonably reliable and efficient. ASAX seems therefore a good candidate to build a powerful DIDS upon.

We are setting up a benchmarking experiment on 24 Solaris machines. We use two Solaris tools, BSM (Basic Security Module) [4] and Netlog [3], to generate the audit data. We then have ASAX analyze the generated audit trails. We assess the cost of the audit analysis with standard Solaris measurement tools, `sar` (system activity reporter) and `perfmeter`. This extended abstract reports preliminary measurements made during one month. Whereas this time interval is too short to be really significant, the results are rather encouraging. The performances of the audited network does not seem to suffer from the IDS. We plan to refine the experiment to have more precise measures on a longer period.

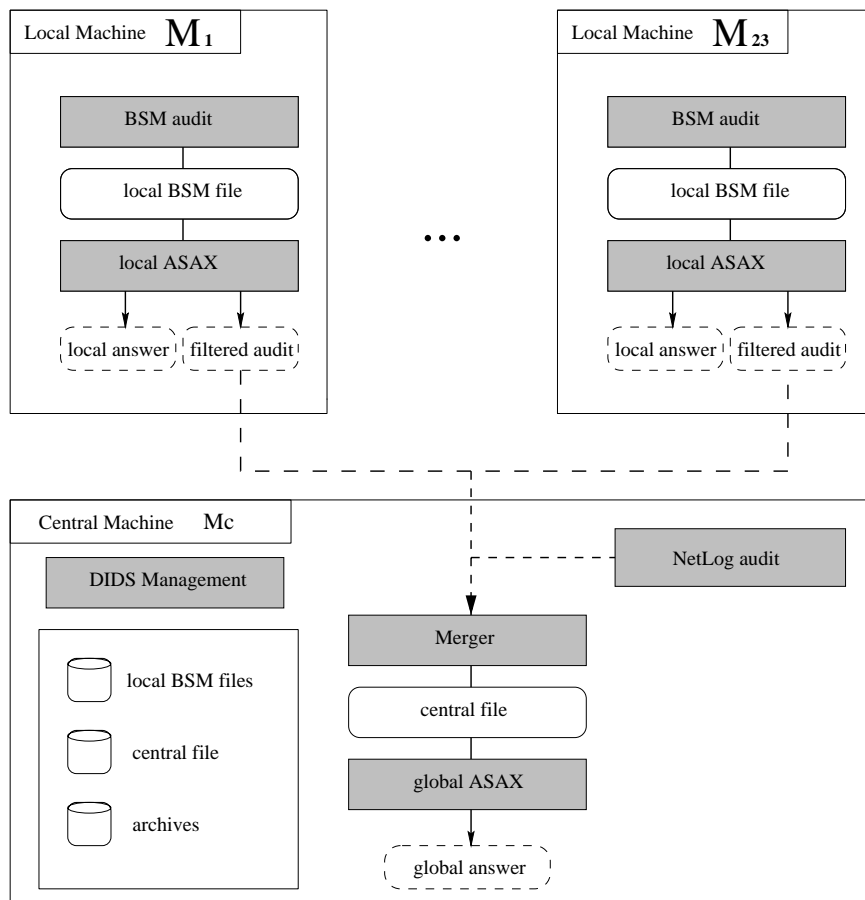


Figure 1: The architecture of our DIDS

In the following, Section 2 describes the architecture of the benchmarked DIDS. Section 3 gives the preliminary results of a short period measurements.

## 2 Architecture of the benchmarked DIDS

Our DIDS audits the Unix machines used for teaching purposes at the computer science department of the INSA of Rennes, namely 23 machines of type SUN SS5 and under Solaris2.5.1. They are connected via a 10 M-bits/s Ethernet. An additional machine, a SUN SS20 server under Solaris2.5.1, not used by students, handles the audit data storage and the centralized handling of part of these data.

Figure 1 shows the global architecture of our DIDS. On every machine ( $M_i$ ), a BSM job generates events reflecting the system activity of this machine. These events give, in particular, a thorough trace of the actions of

the connected users. BSM stores these events in a file. In order to reduce the amount of stored data, BSM is configured to keep information related to processes, system administration and user connections, only. A local ASAX module analyzes the BSM events of the file in order to detect local attacks. The analysis is done on the fly, but the file is not, yet, pruned when the events have been used. When a signature is detected, the ASAX module either sends a report or takes action to limit the impact of the suspected users.

Some of the signatures cannot be detected on a local machine. For example, if an intruder tries to connect once or twice on all the machines of the network, it will pass undetected by the local analysis. Therefore, the local ASAX modules filter the BSM events and send the ones related to distributed attacks to the central machine ( $M_c$ ). Furthermore, events related to connections are gathered by Netlog. Note that events related to “normal” connections are filtered out. For examples NFS transfer information is irrelevant for intrusion detection.

A Merger gathers all the filtered information coming from the distributed machines and the network events, it produces a single file, where events are ordered. The current order is somehow arbitrary. This issue will be addressed in more detail in the next experiments. A central ASAX modules analyzes the central file in order to detect “global” signatures.

The communication between the different modules is handled via PVM (Parallel Virtual Machine) [1] programs.

The local files are currently stored on the central machine via NFS. The experiment has been set in the middle of a semester and we did not want to take the risk to reorganize all the file partitions while teaching and student projects were going on. We plan to do this reorganization during the summer break.

The DIDS is managed by a central module on the central machine  $M_c$ . Currently, it only consists of scripts to start all the modules. We plan to extend it such that it can dynamically reconfigure the analyses. For example, if a user has a suspect behavior we could want to start a more thorough analysis of his activity. This would require, at least, to reconfigure BSM in order to record more detailed events, and to activate more ASAX scenarios.

**Scenarios.** The following scenarios are running in ASAX modules:

- Trojan horse, detected by executable files of sensitive commands which have unauthorized paths (local detection)
- Attempted break-in, detected by several unsuccessful attempts to connect or to change identity (local and global detection)
- Masquerading, detected by too many identity changes (local and global detection)
- Suspicious network connections, detected by a connection from a black list (global detection)

- Leakage, detected by the consultation of sensitive files (local detection).
- Nosing, detected by numerous moves in the directories and file consultations (local detection)
- Abuse of network privileges, detected by a consultation of file “.netrc” and a remote connection (local detection)
- Exploitation of an old `lpr` flaw, detected by a composed signature: “`lpr`, then a `rm` or a `mv`, then a `mv` (local detection). This security hole is now fixed, it is still interesting to detect attempts using this flaw (some hackers may not be up-to-date, they are still potentially dangerous). Furthermore, to test the system it was important to have signatures on several events.

This set of signatures is far from being complete. It covers however, a number of different types of signatures for example black lists, sets of commands, sequences of events. Furthermore, most scenarios can detect several variants of signatures. For example, the detection of attempted break-ins can take a set of alternate commands into account. This set of scenarios is therefore already useful to test ASAX functionalities and performances.

### 3 Preliminary results and discussion

**Performances.** Measurements have been taken in two phases, firstly without the DIDS during a month and secondly with the DIDS during another month. Both months had normal teaching and project load (ie no school holidays). We have used the `sar` (system activity reporter) provided by Solaris to measure the system activity of all the machines. The `perfmeter` tools, also provided by Solaris, has been used to measure the network activity of the central machine  $M_c$ .

Table 1 gives the averages of the measurements. The measurements have been made every day, including week-ends, between 8am and 1am next day. Students do not have access to the machines between 1am and 8am, they however have access during the week-end. For the central machine the given measurements correspond to averages per day. For the distributed machines  $M_i$  the daily averages have been in turn averaged by the number of machines.

A thorough analysis of the benchmark will be made once a longer and more precise experiment has been run. In particular it will be mandatory to analyze the picks of activities.

We can already notice that, on average, the performances measured on the distributed machines  $M_i$  are roughly the same with or without the DIDS. This is already an encouraging sign.

On the central machine  $M_c$ , the CPU load is almost doubled but stays at a very reasonable level. Disk occupation is below 1% which is more than acceptable. Some disk transfer is noticeable for the audit disk. This was

<b>Local machine (<math>M_i</math>)</b>	Without DIDS	With DIDS
CPU utilization (%)	8.34	8.56
System disk occupation (%)	0.71	0.76
System disk transfer (Bytes/sec)	6288	6488
<b>Central machine (<math>M_c</math>)</b>	Without DIDS	With DIDS
CPU utilization (%)	0.9	1.59
System disk occupation (%)	0.05	0.06
Audit disk occupation (%)	0	0.62
System disk transfer (Bytes/sec)	916	850
Audit disk transfer (Bytes/sec)	0	3190
Number of packets/sec	3.54	191.20

Table 1: Average daily performances, with and without DIDS, of the central and distributed machines

<b>File</b>	<b>Size (k-Bytes)</b>	<b>Compressed size (k-Bytes)</b>	<b>Events</b>
Central	2320	255	2250
Local	475	55	5222

Table 2: Daily average size of collected files

predictable as all the local audit data are currently stored on the central machine. However, the rate is about 4 K-Bytes per second which is still several orders of magnitude less than the nominal rate of the Ethernet network (10 Mega bits per second). The number of packets per second has been multiplied by 50. We expect that storing the local audit files on the local file system of the distributed machines will significantly reduce this traffic.

**Size of Audit data.** Table 2 gives the average size of the audit files per day. It also displays the average number of parsed events. Every day around 14 M-Bytes of audit files are stored. As the current file systems can generally cope with several Giga bytes, the amount of storage required everyday is acceptable.

Keeping the audit data could become a problem. The size of audit files once compressed by gzip is, however, divided by 8 as shown by Table 2. On the distributed machines, BSM files are compressed and archived 4 times per day during 4 days. A new archive overwrites the previous ones. The average size of this archive is 5 M-Bytes. Currently, the central file is not yet archived. It has to be done.

**Detected attacks.** During the month of experimentation, our DIDS has detected:

- 6 attempted break-in on the console of 2 machines,
- 1 attempted break-in distributed on several machines,
- 2 nosings on one machine.

We have also acted as intruders in order to test our implemented scenarios. All the scenarios have detected their corresponding attacks, whether on one event or several; intertwined attacks have also been detected without problems. The DIDS could resist an attempted break-in by program. The machine load returned to normal as soon as the attack was over. The most important problem came from the number of generated e-mails and reports, there has been somehow a chain reaction. We have to re-shuffle the reporting mechanisms.

## 4 Conclusion

We have measured the average costs of running, for a month, a distributed intrusion detection system based on ASAX. We have to refine the experiment, in particular to add more scenarios and to analyze the pick of activity of both casual users and intruders. The current result are, nevertheless, very encouraging as the CPU is on the average not perturbed and the size of stored data is quite reasonable.

**Acknowledgments** Jérôme Allard installed ASAX at the INSA de Rennes and made informal preliminary experiments on a single machine. He and Sébastien Blaisot set up an initial ASAX DIDS system. Isabelle Puaut made fruitful comments on an earlier version of this abstract.

## References

- [1] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press, 1994.
- [2] A. Mounji. *Languages and Tools for Rule-Based Distributed Intrusion Detection*. PhD thesis, Faculté Universitaire Notre de la Paix de Namur, Belgium, September 1997.
- [3] CIS network group. Netlog web page, Texas A&M University. <http://www.net.tamu.edu/network/tools/netlog.html>.
- [4] SunSoft. *Solaris SHIELD Basic Security Module*, October 1993.