Implementing Security and Control in Computer Networks

Leandro Márcio Bertholdo *Liane M. R. Tarouco*

Instituto de Informática
Universidade Federal do Rio Grande do Sul
Av. Bento Gonçalves, 9500 - Campus do Vale
91509-900 Porto Alegre, RS
E-mail addresses:
berthold@penta.ufrgs.br liane@penta.ufrgs.br

Abstract

The present paper presents the CUCKOO[1] system, a proposal of a new philosophy in terms of the security of systems. The CUCKOO system uses network management concepts such as SNMPv2, and security concepts such as DES and MD5, along with the implementation of a new set of tools that guarantee the integrity of the systems involved. The result is a system that warns about attempts of attacks and situations of risk to which the many machines under its control are exposed to, thus increasing the level of control that the administrator has over the security of the network.

1. Introduction

After some years of connection to the Internet, the Federal University of the State of Rio Grande do Sul, Brazil, began to realize that, along with the benefits of being directly connected to the scientific community worldwide, it was also exposed to an ever growing hostile environment. This opening has put the institution in a delicate situation, in which frequent problems related to the security of information and the operation of the machines have shown the fragility in face of the attempts made.

The community worldwide has defended itself as much as possible. A number of tools are devised yearly, such as SATAN [VEN95] [BER96], ISS [CER93], PINGWARE [BEL96], and AutoHack [MUF95], intending to realize an audit on the security of machines or subnetworks; other tools are devised to detect any invasion occurred in the past, and others, still, try to guarantee the security through filters, as TCPWrapper [VEN92]. Another way of maintaining the security of a site that has also been used is the isolation of the machines connected to the Internet, such as the various techniques of *Firewall* [CHE94] [CHA95].

After exhaustive studies and follow-ups of the existing vulnerabilities and the ways to explore them, we noticed the lack of a device able to capture the exact moment of an invasion. To know the exact moment of an invasion is extremely useful so that the administrator can take the necessary steps in an efficient manner, and in a crucial moment. In practical situations, it has been many times noticed that a late recognition of an invasion (after one or two hours) may turn the situation into one that

[1] Analogy made with the mechanism of the wall clock that announces an event, that is, the changing of hours.

cannot be controlled and may compromise many other sites in a permanent and non-traceable manner.

The creation of the CUCKOO system has as main goal the filling of this gap, alerting the local administrator to the identification of alterations that may somehow compromise the security of the system. It aims at enabling the administrator to take notice of possible attempts of violation of security, as well as trying to discover which techinique has been used in a given invasion, and to enable some emergency actions in order to prevent the intruser from taking over the system; thus, allowing the necessary time for any flaws in the security to be corrected.

At first, the system was designed to identify and alert the administrator in charge of the netwotk security in the cases of exposure or suspicion of invasion in one of the machines. Among the problems chosen to be alerted of are the following:

- One of the machines had its operational system or applications compromised by tools with *rootkit*[2].
- A host is collecting passwords in the network with a program of the *sniffer* type.
- A host registered access from an unknown place.
- One of the configuration files was altered, as, for example the password file itself.
- There has been a register considered "atypical" in the system logs.

Besides the above mentioned itens, the system is quick and easily used, and does not require from the administrator much time and effort spent in the monitoring of the machines. The application also enables the user to obtain information in a fast and consistent manner in case of an invasion. The purpose of this information is to try to identify the source and the cause of a possible attack.

2. The CUCKOO System

Aiming at the mitigation of the problem of an open system, directly connected to a computer network, researches were carried out in order to develop the automatized detection of intrusions[3]. These mechanisms, totally automatic, aim at defining attempts of violation in the security of a given machine, and, in doing so, revealing new flaws, as the exploitation of new bugs, buffer overflows, among others. Some endeavors in this area use artificial intelligence techniques, and, today, encompass basically two groups [SPA93]:

- the detection of anomalies;
- the detection of violation.

The CUCKOO system uses the first of the two approaches. The detection of anomaly is based on the assumption that the intrusive activity often manifests itself as an anomaly. This approach relies on

---

[2] The RootKit is a set of tools used by hackers that substitutes several applications such as z2, es, fix, sl, ic, ps, ns, ls, and others, to facilitate the entering in the system and to hide traces of invasions.

[3] Conceptually, *intrusion* is understood as the detection of an illegal activity and the acquisition of priviledges that cannot be detected through the normal flow of information and the models of access control.

constant measures made by the system, so that it detects an intrusion through the great variances in these measures. An example of this is the great number of connections that are rejected by the time-out system, a possible evidence of an attack by service denial (synflooding).

To detect an anomaly, the system maintains a knowledge base about different operational systems and users, and uses it to compare to the present state of a machine. This will be further explained in the following chapters.

The detection of an intruder is a new security approach. It provides a sense of security in a network of computers and data, while it allows these to operate in an "open" manner. The aim of this technique is to identify – preferably in real time – the unauthorized use, the misuse, and the abuse of computer systems by both internal users and/or external invaders [MUK94].

The second approach, the detection of violation or abuse, is based on specific techniques of representation of knowledge about "unacceptable behaviors", and attempts to detect these occurrences. An example of this would be the detection of abuse of fingerd and sendmail, used in the attack of "Internet Worm" [SPA88].

2.1 The Structure of the CUCKOO System

The structure of operation of the application uses a philosophy of monitoring in which a single manager monitors, via SNMPv2, the measures of tens of hosts (figure 1).

Each host has an SNMPv2 agent, that communicates with the manager, passing on to it possible alerts of violation, or responding to consultations on the state of its objetcs. The normal consultation is a way of verifying the activity of the agent, since the opposite can mean that an attack is in progress. In other words, all the security added to the system depends basically on the reliable communication among the hosts monitored in a manager program running in the directing machine.

The struture of the application is formed by two basic modules that implement, besides the management protocol, other functions:

Monitor Module: Composed of various application programs, responsible for the identification of a possible invasion.

Director or Manager Module: Composed of a friendly interface that allows SNMP consultations and the possibility of notification of any anomaly to the administrator responsible for the security (security official) of the site, so that he or she take the necessary steps.

The exchange of messages between the manager and agent modules is realized using the SNMPv2 protocol. This choice is due to its features incorporated from the S-SNMP protocol, what guarantees the confidentiality and authenticity of the messages exchanged between partners.

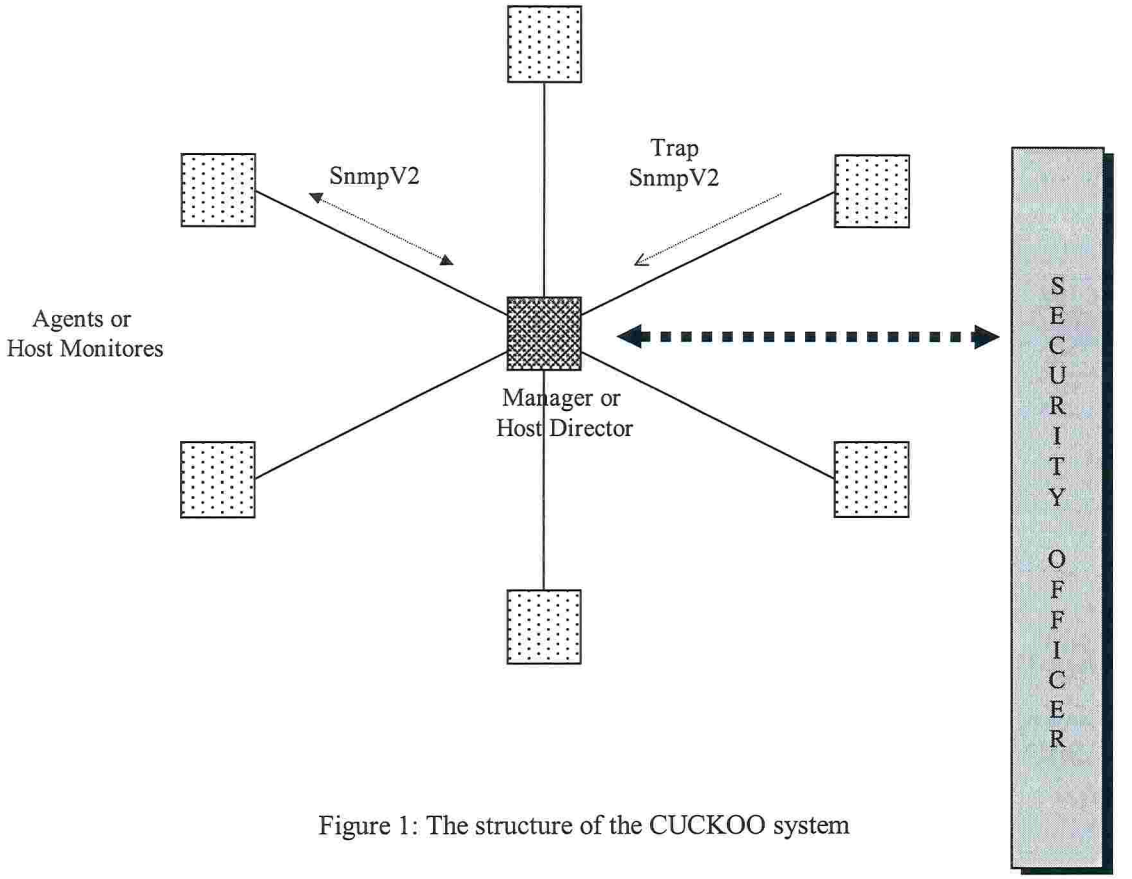

Figure 1: The structure of the CUCKOO system

2.2 The Monitor Module

The monitor module is the core of the CUCKOO system. It is directly in charge of the security of the host where it is installed. It is up to it to detect any type of attack or modification in the config files or even the installation of trap-doors [GAR94], both in the operational system and in the applications. The monitor guarantees the integrity of the user's sensible data, besides keeping an information base (MIB) about the state of objects, activating traps whenever needed. The monitor module has five equally important attributions, which are thus distributed:

1. The implementation of several modules capable of recognizing the occurrence of an invasion: integrity checkers of the operational system and of the system's configurations, monitor of the system's connections, controller of the status of the interfaces of the hosts' network.

2. Automatized control of the system logs, which obtains a greater control over the status of the system using, for this purpose, programs that search automatically the existing logs in the machine for traces.

3. The implementation of the manager protocol, the SNMPv2 agent.

4. The implementation of a module capable of providing information, from the machine attacked.

5. Finally, a module in charge of the gathering of further information on the activities under suspicious that are being carried out by a user of the system. This module must be composed of tools that implement modifications in the codes of the shell program and of a monitor at the user's terminal level (tty), as well as various scripts that allow for the extraction of further information from the system. This module has been partially implemented, since one already finds in literature applications that implement monitors of tty [BELa] and modified shells, thus demanding only adaptations so that it can function directly with the interface of the CUCKOO system.

The figure represented below shows the interrelationship of all the existing submodules that operate together to allow for the above mentioned functions to be realized.

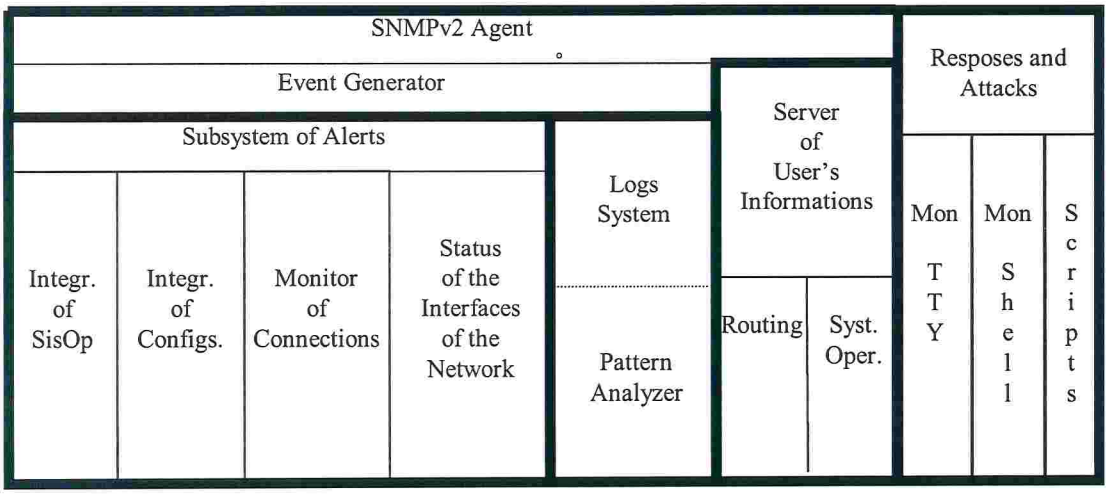

Figure 2: The structure of the monitor module

2.2.1 The Subsystem of Alerts

This subsystem is composed of the procedures of integrity of configurations and of the operational system, connections' monitor, and the checker of interfaces, functions that form the real system of alerts of the CUCKOO application.

The alerts generated by this subsystem are based on a regular verification of the config files and binaries of the operational system, as well as a general view of the status of the system at given times.

2.2.1.1 The Procedures in the Evaluation of Integrity

The evaluation of the Integrity, both of the operational system and the configurations of the system are made using the MD5 algorithm. This evaluation is accomplished in two levels, one considered mandatory (binaries of the operational system), and another considered optional (several files of configuration and/or data). The results obtained with the execution of the algorithm feed directly the

different objects of MIB SNMP. Regularly the manager consults the data of all the agents in its jurisdiction, for a confrontation with a database existing in the host director.

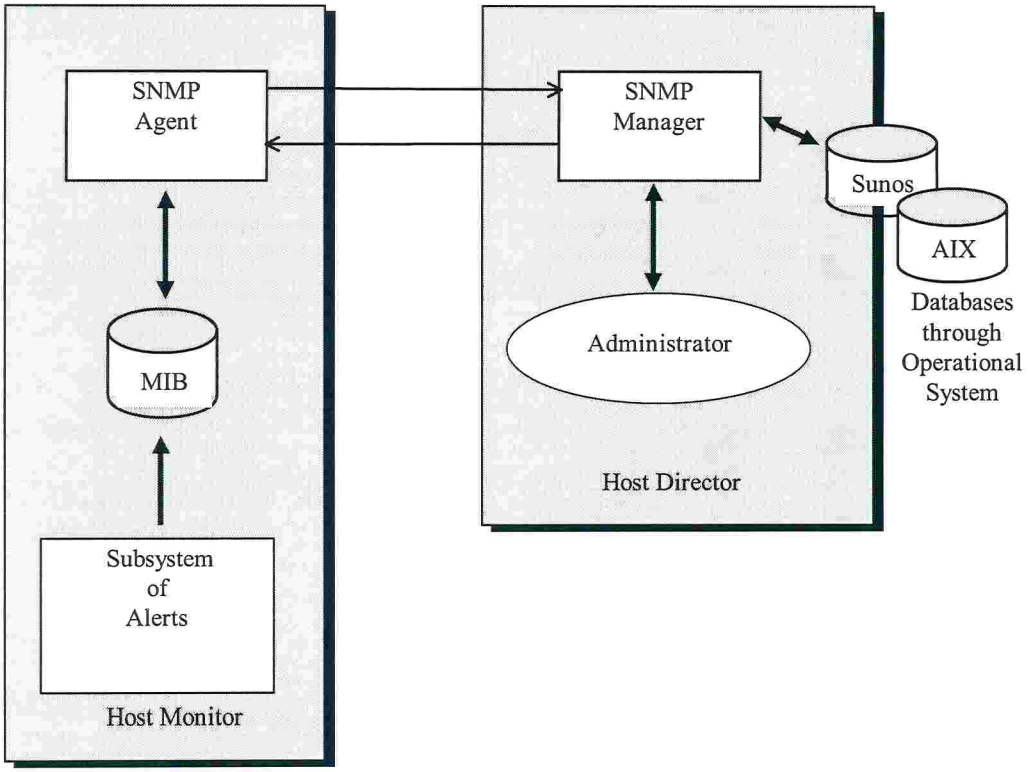

Figure 3: The Integrity Modules

In the case of the verification of the integrity in the Operational System, the manager compares the value of the objects of MIB provided by the agent, based on local data, where the MD5 of some applications considered vital by Unix system can be found. If the verification is related to a system's config file, a database organized by the monitored host will be consulted, where an index with the MD5 of all files of configuration considered important can be found, as well as a copy of these files, so that the alteration realized can be verified.

The maintenance of the copies of the files of configuration is not an automatic procedure of the system, and it is a task of the administrator to keep them updated. This procedure can be fairly automatized, but this is done at the administrator's own risk and cost. Although, at first, it seems unfeasible, this procedure is not so arduous, and is most valuable. In the two places where the system was tested, changes in the configurations are unusual, and, in case of an invasion, are most useful for the verification of alterations[4].

[4] Not always the alterations realized in the config files of the system are easily noticeable; as is the case of the removal of a single character, that can leave the site totally exposed. This sort of subtleness was observed in one of the invasions detected.

The present implementation of the prototype has some binary programs that were chosen to be under constant monitoring (table 1), in addition to some files of system configuration (table 2).

SunOs 4.1.3	AIX 4.1.4
/bin/login	/usr/bin/login
/usr/kvm/ps	/usr/bin/ps
/usr/etc/in.telnetd	/usr/sbin/telnetd
/usr/etc/in/ftpd	/usr/sbin/ftpd
/usr/etc/rshd	/usr/bin/rshd
/usr/etc/rlogind	/usr/bin/rlogind
/usr/etc/inetd	/usr/sbin/inetd
/usr/bin/passwd	/usr/bin/passwd
/usr/bin/su	/usr/bin/su
/usr/etc/ifconfig	/etc/ifconfig

Table 1: Binaries monitored by the system

SunOs 4.1.3	AIX 4.1.4
/etc/passwd	/etc/passwd
/etc/rc.local	/etc/rc.tcpip
/etc/hosts.equiv	/etc/hosts.equiv
/etc/hosts.allow	/etc/hosts.allow
/etc/hosts.deny	/etc/hosts/deny
/etc/inetd.conf	/etc/inetd.conf
/etc/syslog.conf	/etc/syslog.conf
/etc/sendmail.cf	/etc/sendmail.cf
/etc/services	/etc/services
/etc/aliases	/etc/aliases
$HOME/.forward	$HOME/.forward
$HOME/.rhosts	$HOME/.rhosts

Table 2: Files of configuration monitored by the system

Object to be managed	Interpretation
/etc/hosts.equiv $HOME/.rhosts	Alterations allow for the access to the system without authentication
/etc/ifconfig	Possibility of occulting eavesdropper in the network
/vmunix ou correlato	Alteration in the kernel of the system for the installation of trap-doors
/etc/passwd	User with special priviledges, or facilitation for inappropriate access
rc.local, inetd.conf, services ou correlatos	Alteration in the type of service of network made available
/etc/syslog.conf	Attempt to hide the passage through the system
Arquivos de diretórios /bin /usr/bin /sbin	Possibility of installation of trap-doors

Table 3.: some events capable of generating alarms

Above listed one finds some of the verifications carried out by the system. Logically there are others that are intrinsic, such as the agent responsible for the verifications and its data tables, both inserted in the code of application, which is verified whenever the program is executed. All verifications realized by the system are based on hashing functions, more specifically in the MD5 algorithm [SCH94]. The existing implementation has also different levels of configuration, in which there is a minimum level, a maximum level, and one that the user can customize. This level refers to which files the recognition of authenticity will be extended to.

2.2.1.2 The Monitor of the System's Connections

The monitor of the system's connections is also one of the components of the subsystem of alerts. Fistly, threats like SYN flooding are identified by it. The information on the status of the TCP connections is already an object provided by MIB SNMP.

The object of MIB that is analyzed is TcpConnEntry, which provides an overview of the TCP connections, and the present status of each one. In case of more than five requests of connection waiting in line to be attended (synReceived status(4)), the CUCKOO system considers the situation as an attack against the system.

Information on the status of the connections are vital in order to confirm suspicions of invasions that may be in progress, or even that this particular host is being used as a scaffold for the attack against other sites. Among the information obtained through this type of consultation, one finds: the number of the port number on each side of the connection and the IP address of the two involved parts, besides the number of available bytes in the vectors of message sending and receiving. The CUCKOO system analizes this information through a script in PERL and, in case of suspicion, passes the information obtained on to the security administrator of the network.

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address Foreign Address       (state)
tcp        0     44  circulo.nntp  jukebox.plug-in..1051 ESTABLISHD
tcp        0      0  circulo.nntp  lakesis.fapesp.b.2524 ESTABLISHD
tcp        0     80  circulo.telne porta3.tche.br.1024   ESTABLISHD
tcp        0      0  *.*           *.*                   CLOSED
tcp        0      0  *.smtp        *.*                   LISTEN
tcp        0      0  *.writesrv    *.*                   LISTEN
tcp        0      0  *.smux        *.*                   LISTEN
tcp        0      0  *.nntp        *.*                   LISTEN
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2342 SYN RECV
```

Figure 4: Connections of a machine under attack to the WWW service

The analysis of this command allows the detection of attempts of attack using SYN flooding techiniques. Depending on the implementation that is being used by the agressor, the monitor program can locate its origin and generate a report on the machine that is being used as source of the attack, and, if possible, obtain information in relation to the person who is practicing this aggression, besides the generation of other information such as who is accountable for the site or the contact person.

2.2.1.3 The Controller of the Status of the Network Interfaces

The control over the status of the network interfaces is viewed as fundamental to the CUCKOO system. This verification avoids further damage caused by hackers that may break into a host located in a critical point of the network, that is, in one of the main backbones. The basic purpose of this subsystem is to avoid the realization of a monitoring of the traffic in the network (eaversdropper), and, through this, obtain names of users and their passwords in other machines, located in better protected points of the network, or even in other sites.

The CUCKOO system carries out this control through the maintenance of an experimental object in the SNMP database. The object created is called "ifStatus", and is a complement to the "ifAdminStatus" object, already existing in the MIB. This new object is further detailed, having a set of values that differ from the one existing today. It permits a better specification of the status of a certain interface. In this new set of values valid for the object, one finds broadcast(4), debug(5), and, specially, promiscuous(6). Through it, the host director can assess the occurrence of something suspicious in a given station. This option needs to be disabled in hosts that use RMON agents or that run programs that execute monitorings in the network, such as TCPDUMP, since, otherwise, these machines will always be considered "under attack" by the CUCKOO system.

The assessment of the status of the interfaces is realized by a function called CheckStatus(). This function uses the same resources as the ifconfig command does to obtain a list of the interfaces existing in each host and its status. It is through this function that the ifStatus object of MIB SNMP is maintained.

```
le0: flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>
        inet 143.54.1.20 netmask ffffff00 broadcast 143.54.1.0
lo0: flags=49<UP,LOOPBACK,RUNNING>
        inet 127.0.0.1 netmask ff000000
```

Figure 5: Information on the status of the interfaces obtained by CheckStatus()

2.2.2 The System of Logs and the Pattern Analyzer

The logs of activities of the system are, in general, precious information to any administrator that knows how and wishes to make good use of them. The data analysis module involves a set of programs responsible for the generation and handling of the information provided by the operational

system and the applications, such as httpd, telnetd, ftpd, logind, rlogind, rshd, tcpd, and xtacacsd, and others that the administrator wishes to their outcome analyzed.

This subsystem is based on the search for standard messages of the system, as the ones shown in Figure 6, as well as many others that may reveal traces of attempts of subverting a given host, or that are not considered by the administrator as routine. It is interesting to point out, however, that the system will notify all messages not considered harmless by the administrator. This module can be used independently from the manager module.

This submodule was designed to execute and verify automatically the files generated by the logs system, locating violations or unusual activities realized against the system. The logs analyzer uses for this a database of filtering rules for the logs generated by the operational system and applications.

The application code is subdivided in two programs that act identically: one that enables a general analysis of the logs system, thus generating a report about all "infractions" that were registered in the different applications; and another in charge of the simultaneous analysis of logs generated in each monitored host.

The database of rules ir formed basically by three files: one for the identifications of what is considered a violation by the system (log.violation), another to identify possible important warnings registered by the system, such as reboots, daemons reestarted, etc. (log.warning), and, still, another that specifies everything that is to be ignored by the system (log.ignore).

In the log.violation file there are registered verified patterns that certify more than 90% probability of an attempt of hacking of the system, as can be verified in the following file extract.

```
mailto:root     "wiz"
mailto:root     "WIZ"
mailto:root     "debug"
mailto:root     "DEBUG"
mailto:root     ATTACK
mailto:root     nested
mailto:root     VRFY bbs
mailto:root     VRFY decode
mailto:root     VRFY uudecode
mailto:root     VRFY lp
mailto:root     VRFY demo
mailto:root     VRFY guest
mailto:root     statd.*attempt to create
mailto:root     .bin.+sh
```

Figure 6: Extract from the log.violation file

In the log.warning file, any enterings of log that deserve special attention from the system's administrator are specified. This was the main idea that generated this submodule: to reduce the amount of log to be inspected by the administrator.

```
warning         FAILURE
warning         REFUSED
warning         BAD
```

```
warning         permitted
warning         PERMITTED
warning         rexec
warning         illegal
warning         ILLEGAL
warning         courtney
warning         ATTACK
warning         natas
warning         SATAN
warning         setsender
warning         securityalert
warning         nested
warning         sucked
warning         -ERR Password
warning         !=
warning         SITE EXEC
warning         RETR group
warning         RETR passwd
warning         RETR pwd.db
warning         CWD etc
warning         named\[[0-9]+\]: approved AXFR from .+ for
```

Figure 7: Extract from the log.warning file

In the log.ignore file one finds inserted all the words that identify not so important enterings registered by syslog. As the configurationi of the facilities of the syslog.conf file used, this file became long, registering all that was possible, so that no message till then known would go unnoticed. This procedure is necessary in order to identify possible violations realized agianst applications such as ftp, sendmail, and the name service (DNS).

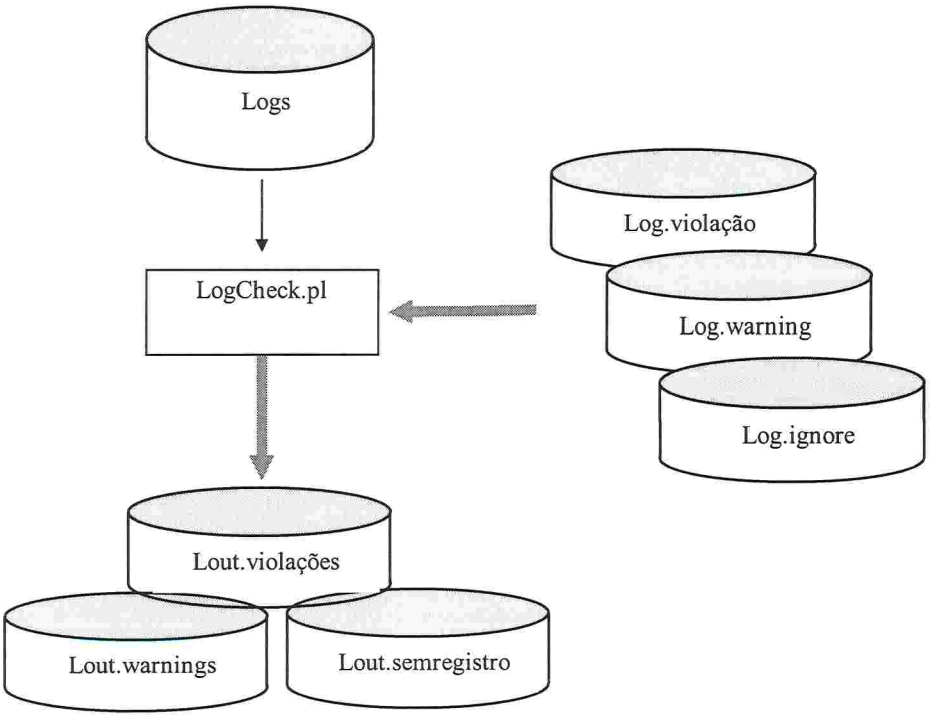

Figure 8: The structure of the logs analyzer

The logcheck.pl program (Figure 8) processes the system logs in a complete manner, that is, all data that are found in the configured logs files (directory /var/log) are verified. As a result, three reports are generated with its conclusions:

- Lout.violations: All the attempts of violation, according the rules configured in log.violations, that is, clearly considered an attack. The existence of this file after the processing of logs indicates that a host monitor, or the host director itself, has been, or is, under some kind of attack.
- Lout.warning: All relevant log information that the administrator has to inspect daily, or in short regular intervals.
- Lout.unregistered: Report of unknown or inexpected entries in the system logs. During the adaptation period of the CUCKOO system to its network, this type of message is common, and has to be frequently revised, in order to fit such patterns in one of the files of the existenting rules.

In its second form (AnLog.pl), the logs analyzer is configured in the host monitor through a call to the system (tail -f $logfile) to monitor, in real time, the occurrences passed on by daemon syslogd. This way of obtaining logs was chosen for its simplicity and portability, in relation to other tests in which changes in programs like syslog, or even syslogd itself, were proposed. Its disadvantage is that the system gets vulnerable to an attack to the daemon syslogd.

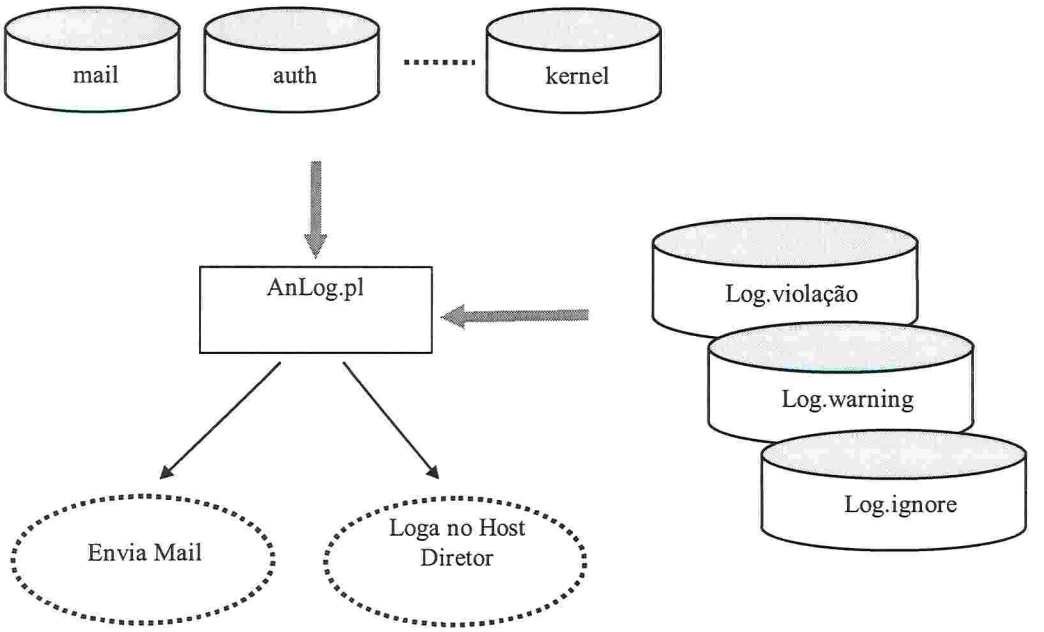

Figure 9: The logs analyzer in the host monitor

The logs analyzer in the host monitor has a different way of registering the problems reported to the host director. The most used way nowadays would be the descentralization of all logs generated by all machines, what would generate an unnecessary traffic on the network, besides the overloading of the system of files of the host director and the demand of a computational effort multiplied many times. To solve this problem, we opted by the creation of the AnLog.pl program that runs in the host monitor, and registers, via syslogd itself (local7) of the host director, only the enterings that it considers necessary (rules of the log.warning file). A second option the system has is to give a special treatment to all cases that combine with the rule described in log.violation: to send a mail to the administrator responsible for that particular host monitor (Figure 10).

```
From root Fri Jan 10 14:40:26 1997
Received: from penta.ufrgs.br (penta.ufrgs.br [143.54.1.20]) by
circulo.pop-rs.rnp.br (8.7.5/8.7.3) with SMTP id OAA16072 for
<berthold@circulo.pop-rs.rnp.br>; Fri, 10 Jan 1997 14:40:25 -0300
From: CUCKOO@penta.ufrgs.br
Received: from circulo.pop-rs.rnp.br (circulo.pop-rs.rnp.br
[200.132.0.21]) by penta.ufrgs.br (051895/8.6.11) with SMTP id OAA14330
for berthold@circulo.pop-rs.rnp.br; Fri, 10 Jan 1997 14:40:55 -0300
Date: Fri, 10 Jan 1997 14:40:55 -0300
Message-Id: <199701101740.OAA14330@penta.ufrgs.br>
To: berthold
Subject: CUCKOO::ALERTA

*****************************
mail: Nov 22 16:28:04 penta.ufrgs.br sendmail[5911]: "wiz" command from
minuano.inf.ufrgs.br (143.54.7.18)
*****************************
```

Figure 10: Example of mail to the administrator

This submodule of the CUCKOO system also considers two types of logs: the ones that can be readdressed, generated by applications such as TcpWrapper and sendmail that make use of syslog; and those generated by programs such as /bin/login, that are intrinsically local.

The CUCKOO system has 2 **wcheck** and **ucheck** programs, used to guarantee the integrity of audit files like /etc/wtmp and /var/log/utmp, easily adultered by applications like "zap", that come in kits such as rootkit. Below are listed the outcomes of the **wcheck** and **ucheck** programs, created exactly to guarantee the integrity of these audit files. The entries, with the "<NULL?>" comment, were a result of the exclusion of the "test" user, using an implementation of the **wedit.c** program, a similar implementation to the one used by "zap" program. Consultations made to these files by traditional applications (last, for example) are not capable of detecting such differences.

```
user[     ftp] line[ ftp9568] host[mithrandir.ibase]
user[        ] line[        ] host[                ] <NULL?>
user[        ] line[        ] host[                ] <NULL?>
user[ aluno96] line[   ttyp2] host[minuano.inf.ufrg]
user[        ] line[   ttyp2] host[                ]
user[        ] line[   ttyp0] host[                ]
user[lisianeh] line[   ttyp0] host[ tigrao.ufrgs.br]
user[     ftp] line[ftp10007] host[  200.255.253.23]
user[        ] line[ftp10007] host[                ]
user[        ] line[ftp10491] host[                ]
user[        ] line[        ] host[                ] <NULL?>
user[lisianeh] line[ftp10522] host[ tigrao.ufrgs.br]
user[        ] line[ftp10522] host[                ]
user[  lauren] line[   ttyp5] host[     noc.tche.br]
user[berthold] line[   ttyp5] host[  porta2.tche.br]
user[        ] line[   ttyp4] host[                ]
user[berthold] line[   ttyp4] host[  porta2.tche.br]
user[berthold] line[ftp10809] host[         circulo]
user[        ] line[ftp10809] host[                ]
user[berthold] line[ftp10810] host[  porta2.tche.br]
user[        ] line[ftp10810] host[                ]
```

Figure 11: Output of the wcheck program for SunOs 4.1.3

2.3 The Director or Manager Module

The manager module, on its turn, is composed of three basic parts:

1. The interface of the system, all based on hyperdocuments, using a well known navigator such as Netscape; a server for the HTTP protocol implementing some basic guidelines of the protocol; programs generating mixed code HTML and JAVA scripts [FLA96].

2. A SNMPv2 manager, created to negotiate with the various remote agents. Its implementation uses the CMU package.

3.A set of C programs, scripts PERL [WAL96] and shell scripts capable of analyzing and collecting information on a possble source of attack, checking the operation or not of the hosts under monitoring, as well as other data analyzing programs.

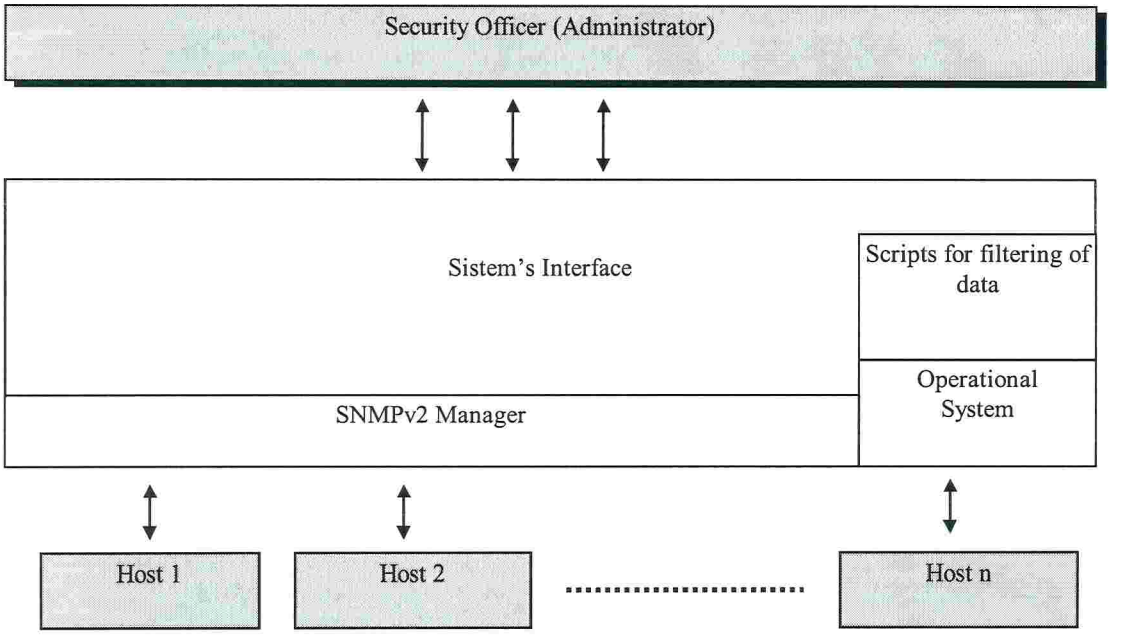

Figure 12: The structure of the director module

2.3.1 The System's Interface

Being part of the objetive of the CUCKOO system its utilization in a variety of machines located in the university campus, the portability of the application for the different existing plataforms is crucial. During the process of design of the system, the portability and consequent simplicity was always taken into consideration[5]. As a result, we opted for the construction of the system's interface using a strategy that is increasingly establishing its grounds: an interface using mutlimedia navigators (browsers). The interface has JAVA scripts [FLA96] embebed in the HTML language, thus allowing the access to data generated by the different application programs and by the SNMP manager.

[5] It was always taken into consideration basic security principles during the design process of the application's code, principles such as KISS (*Keep it Simple Stupid)*.

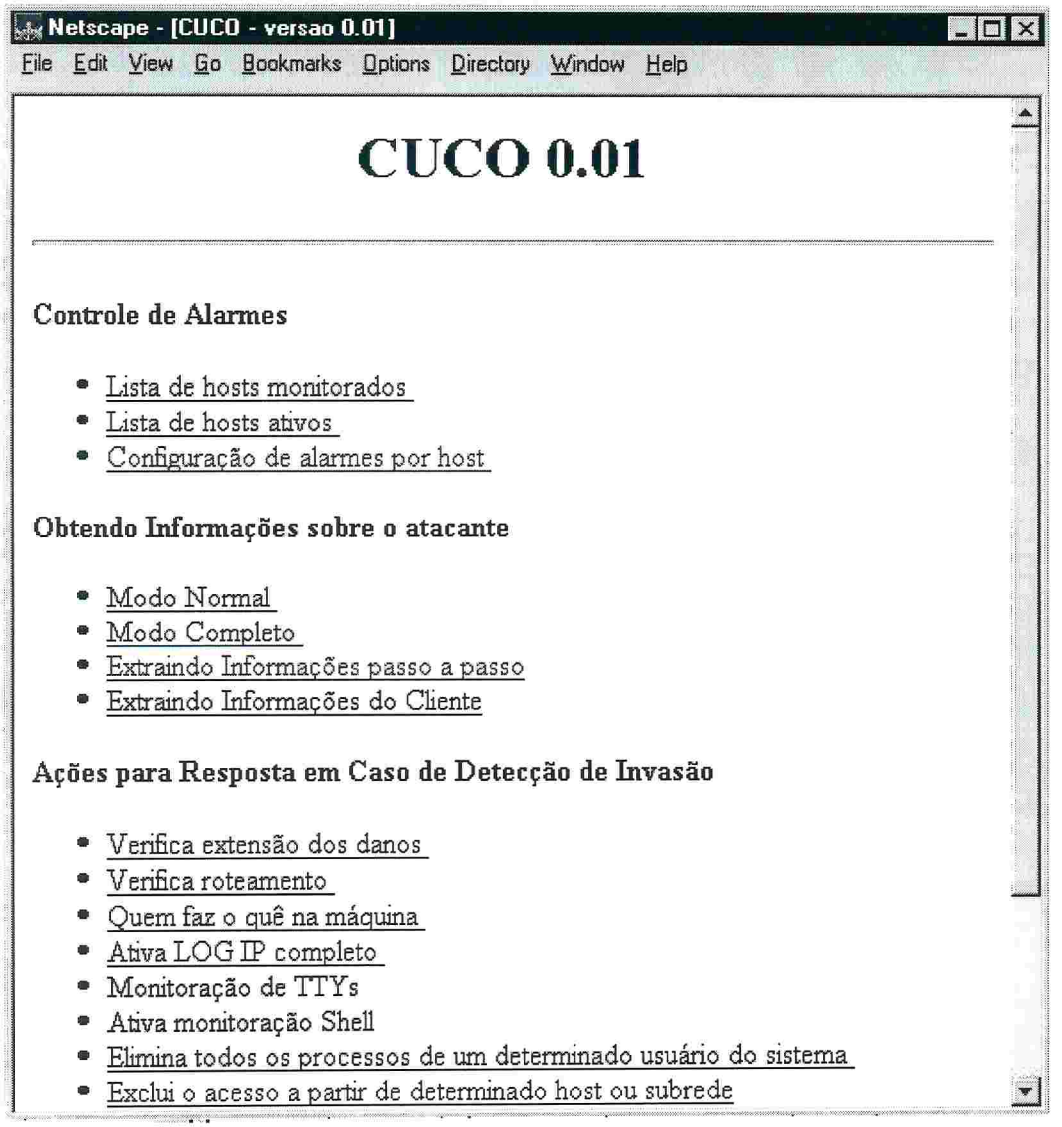

Figure 13: The interface of the CUCKOO system

The system is initially set off by a program called "CUCKOO". From this program, the initial process is dividided via a call to the system (fork()) in two: one responsible for the direct interface with the user, accomplished through the Netscape or Mosaic browsers; another responsible for the control of the HTTP server. Both processes share the same number of communication port previously generated to the processes fork. This way the communication between the user and the HTTP server is maintained.

2.3.2 Locating the Source of the Attack

One of the challenges for the detection of an invasion in a network environment is, beyond doubt, to follow the leads left by the aggressor, these being based on a machine or a person that has

originated the attack, or on a file left behind, keeping in mind that this source can move through many Internet sites during an attack (synchronized attack). For instance, an invader may use different users accounts in different machines during a single attack, aiming at the dissuasion of the network administrator, thus trying to conceal the real source of the attack.

The CUCKOO system has an exclusive module for the detection of the aggressor. This subsystem collects information from many independent sources, including the network status itself and its behavior at the moment of the attack, registering all this information in a file and, later, presenting to the administrator resposible for the security of the network. The identification of a possible aggressor can be accomplished by the person responsible for the security of the network in two different ways, according to the person's criterion:

- A **normal** mode: This mode carries out standard searches for the discovery and investigation of a user in a given host. "Standard" is to be understood as those normally carried out by a network administrator in the attempt to obtain knowledge about a user or the Internet machine, such as nslookup, traceroute, whois, etc.
- An **aggressive** mode: This mode of operation, more complete than the previous one, tries to find out by all means the source of a given connection, in addition to the enabling of the recognition of the machines located in the surroundings of the host aggressor. Taking into consideration the different responses that the system can obtain from the previous methods, the system can recommend to the administrator for he or she to try to obtain further information about the domain at issue through the several machines that exist in between the administrator and the aggressor machine; the former needing a simple mouse selection for a new host to be investigated.

2.4 The Communication Protocol

The protocol chosen to implement the communication and managing of information was SNMPv2 [STA93], since it provides an acceptable level of reliability. The SNMPv1 protocol is not used, since, although widely used in a network management scheme, it has inherent security flaws. Flaws such as the inabily to authenticate the source of the message as well as the prevention of its interception.

Even incorporating some features of S-SNMP, SNMPv2, it still has imperfections intrinsic to the protocol on which it operates, in this case UDP. However, considering the need of security in the communication between the agent and the manager, and, despite still having some problems, the SNMPv2 adapts well to the matter and is able to check:

- The integrity of the data: It eliminates problems such as insertion, duplication, and the ressequencing of messages through a schema of synchronism and time-stamp.
- The authentication of the source: It provides confirmation of the source of a message using a schema of public key.
- The confidentiality of the data: It provides confidentiality through the cryptograhy using a symmetric algorithm (DES).

Among the many modes of operation of SNMPv2, we opted for the "private e authentic" mode, that satisfies all the features above.

Following is a segment from the MIB SNMP that was created to meet the needs of the CUCKOO system.

```
hostDirector OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..255))
              ACCESS  read-only
              STATUS  mandatory
              DESCRIPTION
                      "Informs the host director to which it must report and
have as single manager of the CUCKOO system."
              ::= { CUCKOO 1 }
hostId OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..255))
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
                      "Informs a host that must lose access to the system. The
connections are terminated and new requests are rejected. It is used only if the
CUCKOO system is with the module of response to attack enabled."
              ::= { CUCKOO 2 }
userId OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..255))
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
                      "Informs a user to be eliminated from the system. Used
only if the CUCKOO system is with module of response to attacks enabled."
              ::= { CUCKOO 3 }
ativaTcpDump OBJECT-TYPE
                  SYNTAX  INTEGER {
                          inativo(0),
                          ativo(1)  }
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
                      "Activates dump of packages. Used only if the CUCKOO
system is with the module of response to attacks enabled."
              ::= { CUCKOO 4 }
          ifNumber OBJECT-TYPE
              SYNTAX  INTEGER
              ACCESS  read-only
              STATUS  mandatory
              DESCRIPTION
                      "Number of system network interfaces "
              ::= { CUCKOO 5 1 }
          ifTable OBJECT-TYPE
              SYNTAX  SEQUENCE OF IfEntry
              ACCESS  not-accessible
              STATUS  mandatory
              DESCRIPTION
              "A list of interfaces. The number of entries is given by
ifNumber."
              ::= { CUCKOO 5 2 }
          ifEntry OBJECT-TYPE
              SYNTAX  IfEntry
              ACCESS  not-accessible
              STATUS  mandatory
              DESCRIPTION
                      "Each interface has another few objects"
              INDEX   { ifIndex }
                        ::= { ifTable 1 }
          IfEntry ::=  SEQUENCE {
                ifIndex  INTEGER,
                ifDescr  DisplayString,
                ifStatus INTEGER }
          ifIndex OBJECT-TYPE
              SYNTAX  INTEGER
              ACCESS  read-only
              STATUS  mandatory
              DESCRIPTION
      "A single value to each interface, with values ranging from 1 and
ifumber."
              ::= { ifEntry 1 }
          ifDescr OBJECT-TYPE
```

```
                SYNTAX  DisplayString (SIZE (0..255))
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                        "Textual description of the interface."
                ::= { ifEntry 2 }
            ifStatus OBJECT-TYPE
                SYNTAX  INTEGER {
                    up(1),down(2),testing(3),broadcast(4),debug(5),promiscuous(6)
}
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
   "Interface status, according to the values returned by the fstatus()
function"
                ::= { ifEntry 3 }
```

Figure 14: Segment of the experimental MIB created

3. CONCLUSION

Having in mind the growing number of incidents and the lack of a tool that approaches in a complete manner a specific problem such as the security of machines in an environment clearly vulnerable, the strategy of monitoring proposed is of great woth. The CUCKOO system comes to fill in a gap left by tools as SATAN, for the security proactive checking and others like firewalls and filters, that solve this same problem through access restrictions. This tool is the first to incorporate features auch as alarms and remote security management through SNMPv2, in addition to having features of data analysis and monitoring of users and processes. The tool has a great potential, including the isolation and detection of new forms of attack to the protocols and services available today via Internet.

During the period of time in which the tool was under development, a number of attempts of undue access, or attempts to xplore old bugs in applications such as sendmail, were detected. In attack simulations realized against the system, the application had, in its worst case, only the register of files modification. In this simulation, a buffer overflow had been explored. The application alerted in relation to the attack when there was the attempt of installatioin of a sniffer and a trapdoor in the machine. This avoided the adulteration of the system after the violation, but it was not possible to recognize, through the logs generated by the system, which application was explored.

In other simulations, where our passwords file was transported (common action carried out by hackers), the system registered the output of this file via system logs and reported immediately. This made possible the location of the aggressor machine, as well as the identification of the user that was using it at that moment, via contact with the administrator of the aggressor site.

The CUCKOO system undeniably increased the tranquility with which security problems are dealt with in the monitored machines. It provides a basic guarantee: the operational system is intact, both in its files and in its configuration. The machines are still vulnerable to applications such as sendmail, but we have the guarantee that sooner or later the sources of the attacks will be discovered.

Among the possibilities of expansion in the use of the tool are foreseen improvements such as a monitor for users' terminals (tty) and a monitor for applications such as sh and csh, adapting the already existing tools.

REFERENCES

[BEL96] BELLCORE. **PINGWARE System Master Log Summary.** Http://www.belcore.com/demotoo/SECURITY/pingmaster.html, 1996

[BER96] BERTHOLDO, Leandro; TAROUCO, Liane. **Uma Análise do Software de Segurança SATAN - Security Administrator Tool for Analyzing Networks.** II WAIS Fortaleza - CE, 18-20 de Maio de 1996, pp 149-160.

[CER93] CERT/CC. **Internet Security Scanner (ISS).** Computer Emergence Response Team Advisory 93:14, September, 1993.

[CHA95] CHAPMAN, Brent; ZWICKY, Elizabeth. **Building Internet Firewalls**. O'Reilly & Associates, 1995.

[CHE94] CHESWICK, William; BELLOVIN, Steven. **Firewalls and Internet Security: Repelling the Wily Hacker**. Quarta edição. Addison Wesley Publishing Company, 1994.

[FLA96] FLANAGAN, David; **Java in a Nutshell. A Desktop Quick Reference for Java Programmers.** O'Reilly & Associates, Inc.. First Edition, February, 1996.

[GAR94] GARFINKEL, Simson; SPAFFORD Gene. **Practical Unix Security.** O'Reilly & Associates, Inc. 1991. Sétima edição 1994.

[MUF95] MUFFETT, Alec; **WAN-hacking with AutoHack - Auditing security *behind* the firewall**. Sun Microsystems United Kingdom. The Fifth USENIX UNIX Security Symposium, Salt Lake City, Utah. June 5-7, 1995.

[MUK94] MUKHERJEE, Biswanath; HEBERLEIN, Todd L.; LEVITT, Karl N.; **Network Intrusion Detection.** IEEE Network, May/June 1994, p.26-41.

[SCH94] SCHNEIDER, B. **Applied Cryptography**. John Wiley & Sons, 1994.

[SPA88] SPAFFORD, Eugene H.; **The Internet Worm Program: An Analysis.** Departament of Computer Sciences, Purdue University. Purdue Technical Report CSD-TR-823. November, 1998. Ftp://ftp.cs.purdue.edu.

[STA93] W.Stallings. **SNMP, SNMPv2 and CMIP - The Practical Guide to Network-Managment Standards**, Addison-Wesley, 1993.

[VEN92] VENEMA, Wietse. **TCP WRAPPER: Network Monitoring, Access Control, and Booby Traps.** Mathematics and computing Science - Eindhoven University of Technology. In proceedings of the Third USENIX Security Symposium, Baltimore, Maryland. September 1992.

[VEN95] VENEMA, W.. **S.A.T.A.N. A Summary. Internet Security.** March 1995.

[WAL96] WALL, L.. **PERL Reference Manual - version 5.002beta1g.** January 1996.