USING NAGIOS FOR INTRUSION DETECTION

M. Cárdenas Montes, E. Pérez Calle, F.J. Rodríguez Calonge, CIEMAT, Madrid, Spain

Abstract

Implementing strategies for secured access to widely accessible clusters is a basic requirement of these services, in particular if GRID integration is sought for. This issue has two complementary lines to be considered: security perimeter and intrusion detection systems. In this paper we address aspects of the second one.

Compared to classical intrusion detection mechanisms, close monitoring of computer services can substantially help to detect intrusion signs. Having alarms indicating the presence of an intrusion into the system, allows system administrators to take fast actions to minimize damages and stop diffusion towards other critical systems.

One possible monitoring tool is Nagios (www.nagios.org), a powerful GNU tool with capacity to observe and collect information about a variety of services, and trigger alerts.

In this paper we present the work done at CIEMAT, where we have applied these directives to our local cluster. We have implemented a system to monitor the hardware and system sensitive information. We describe the process and show through different simulated security threads how does our implementation respond to it.

INTRODUCTION

The construction of the infrastructure necessary for the system GRID presents new and interesting challenges. A fundamental aspect to be able to reach the marked aims will be the implantation of an effective system of security. To avoid that the GRID is used by not authorized persons, it will provide confidence to the investigators in his use. In addition, it is indispensable to prevent that the system is used to realize attacks against other systems.

In this context, the intrusion detection systems (IDS) acquires special importance. The intrusion detection systems allow to detect the intruders' presence in the system as soon as possible. This quick detection will minimize the damages produced in the system and avoiding that the platform is used for further attacks to other systems. There are two types of IDS, host intrusion detections systems (HIDS) and network intrusion detections systems (NIDS). A NIDS is a intrusion detection device, which looks at network traffic and tries to detect intrusion attempts based on patterns and specific packets. A HIDS is a intrusion detection device, which seeks for unauthorized changes in files.

There are basically three ways to detect intruders on a system: changes in the filesystem, strange entries in the logfile and strange packets on the network. Our aim has been the study, evaluation and implantation of a HIDS based on Open Source software. A system based on technologies like Nagios, SNMP, Tripwire and Chkrootkit has been implanted in the CIEMAT, in the University of Bacelona (UB) and in the University Autónoma of Madrid (UAM).

NAGIOS

Nagios is a system designed for the monitoring of computers, detection of failures in services and sending notification out to administrative contacts. Nagios is not specifically an IDS. On the other hand, Nagios possesses a friendly interface, is easy to use, very flexible and it is endowed a system of sending alerts.

Nagios has a modular design with a web interface and a set of plugins to check the different services. To point up his ability to support consultations on the protocol SNMP. It can use the check_snmp plugin to check the value of the various OIDs that the administrator is interested in. For this is compulsory that SNMP services are running on the remote host.

There is another way to check local o private services, it is use check_by_ssh. Check_by_ssh is a plugin to execute a script on a remote host using the SSH protocol. Any script it want to execute on the remore host have to be installed on the remote hosts beforehand.

What do we monitor?

As soon as an intruder gains access to a system across a vulnerability, it is frequent that he realizes the necessary actions to conceal his presence and to create a privileged access. These actions can be realized by the installation of a rootkit or manually. In this case, usually the intruder creates an user with privileges of superuser. To detect this action has been created a script to notify the number of users with uid=0 (superuser privileges), sending a alert if this number is bigger than 1.

Less frequent is that the intruder creates a user without password. To detect this anomaly another script has been created.

As soon as the intruder has gained a privileged access to the system, he will try to capture information of other computers on the same network (specially users and passwords). This task will be executed by a sniffer installed by the intruder. The activation of the sniffer will mean that the network interface will be put into promiscuous mode. A

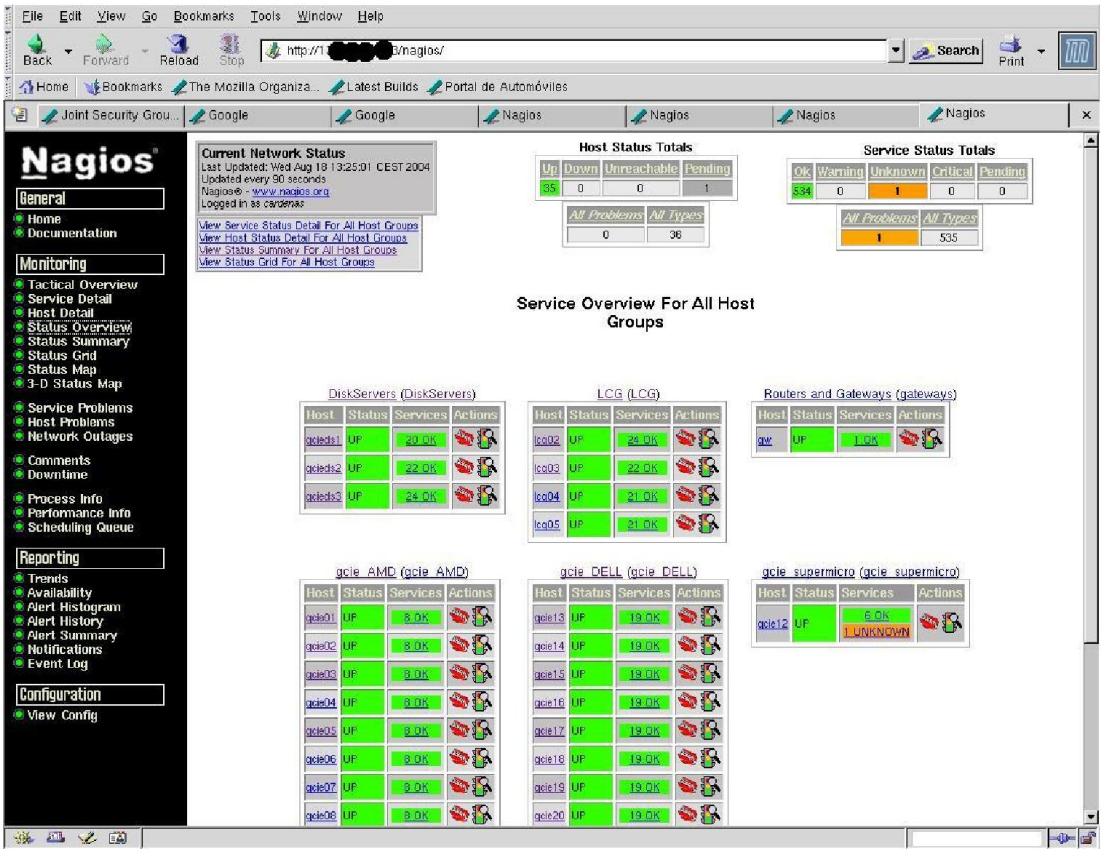

Figure 1: View of Nagios main screen.

script to detect the promiscuous mode in the network interface, also has been created.

Files used by the intruder (binaries of sniffer, configuration files, information captured files) are usually hidden in /dev the directory. Another script has been created to ensure that no regular files have been hidden there.

These four scripts are executed using the plugin check_by_ssh. The information gathered by the plugin is sent to the Nagios monitor. With this set of scripts the sufficient information is covered as to detect quickly the presence of an intruder, so much if he realizes actions to conceal his presence as if not. If an intruder change the ifconfig binary for other one that does not show that the interface is in promiscuous mode, then will not be possible to detect with this command if the interface is in this mode. So it is necessary to prevent that our binaries being replaced into others *trojanized*.

The detection of rootkits and trojans is an aspect not covered by these scripts.

TRIPWIRE AND NAGIOS

A knowledgeable malicious user will try to modify certain binaries of the system. Some of those binaries it will be ifconfig, ls, find, netstat, ps, top... Those binaries modified conceal the signs of presence of the intruder.

For example, the binary ps modified will conceal the execution of the sniffer installed by the intruder. Or in case of ifconfig, it will hide that the network interface is in promiscuous mode. It is in the detection of these alterations of files where the use of tripwire turns out to be strategic. Finally, if the binary ls is altered it will not show the directory where the intruder have installed their files.

Tripwire is an intrusion detection tool able to detect and pinpoint changes to files. In the Open Source version, Tripwire is a command-line tool. On Unix systems, Tripwire is able to detect changes affecting the following properties:

- File additions, deletes and modifications.
- File permissions and properties.
- Inode number and number of links.
- User id of owner and group id of owner.
- File type and size.
- Device number of the disk on which the inode associated with the file is stored.
- Device number of the device to which the inode points.
- Number of blocks allocated to a file.
- Modification, access and creation timestamp.
- Inode creation and modificacion timestamp.
- Hash checking: RSA, MD5, MD4, MD2, SHA and Haval code.

To detect these changes, tripwire establishes a ciphered database of monitored files. Periodically the consistency

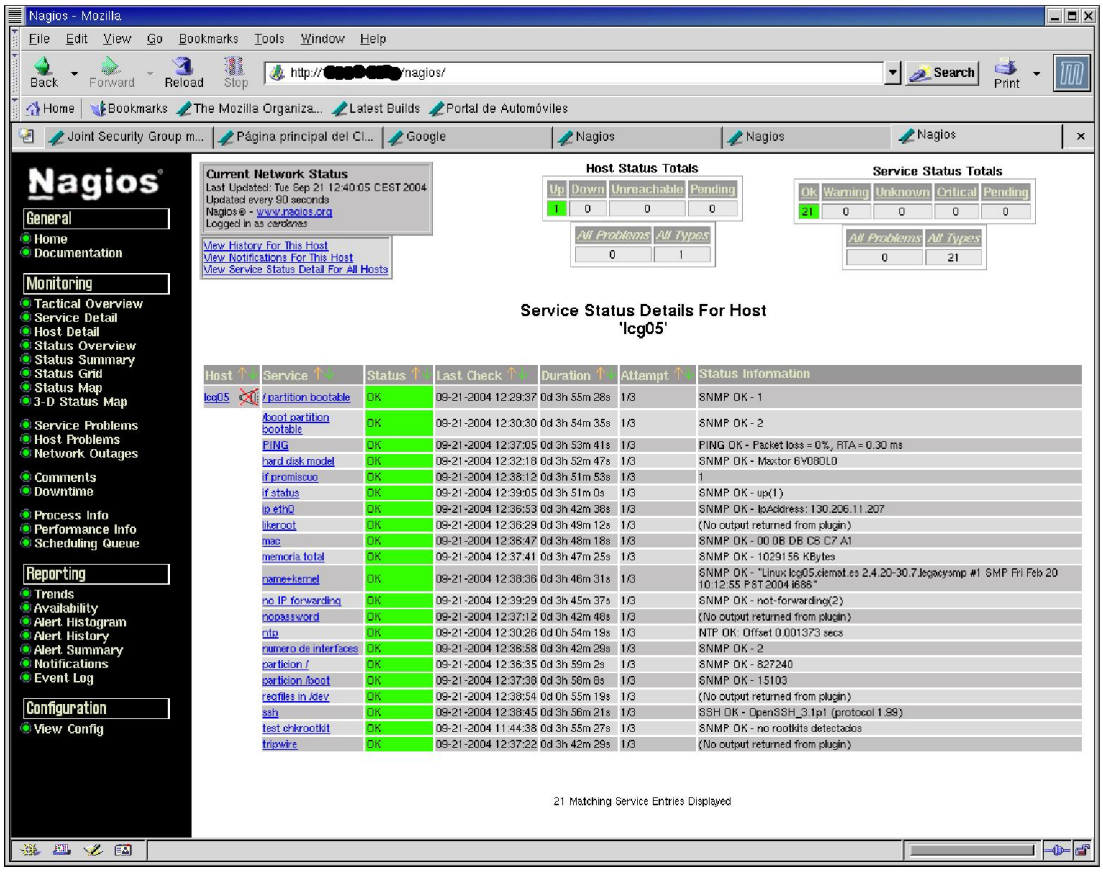

Figure 2: View of a computer services state screen.

of files is checked against the reference information in the database. A report is created with the more relevant information. It is necessary to incorporate the own binaries of Tripwire to the database for assure the self-integrity.

Using the Tripwire database, the administrators can check all the critical files for tampering. Now, how do you know if someone has tampered with yours Tripwire binaries or Tripwire database? After all, if the intruder can modify the Tripwire database, any changes could not be detected.

Several different methods exist. The easiest one is to place Tripwire database on a read-only floppy disk. Since most Linux machines have a floppy drive and few are in use all the time, it's a good match. Other possible schemes include: remote mounting the Tripwire database from another more secure machine read-only (for exemple NFS read-only mount it from a remote, more secure machine with a floppy), putting it on a write-protected Zip disk, or even getting an old, small hard drive that has been jumpered to hardware enable read-only and put it on that. The idea is to put it on some media that you can make read-only in hardware. It does you no good to place Tripwire database where an intruder can mess with it.

At CIEMAT and the other institutes, we have chosen a different strategy. A checksum of database file is executed, and this information is inserted in the MIB tree. The hash is checked by SNMP request against information resident in a central platform.

What do we monitor?

To analyze routinely the consistency between the monitored files and the stored information in the base of information, a script has been created that is thrown for Nagios. This script initiates the execution of triwpire, analyzes the generated report, and sends the resultant information to agios. Based in this information Nagios generates the necessary alerts.

In order to avoid that the execution of tripwire monopolizes too many resources, the checking has been restricted to a few binary of the system. These binaries have been chosen for being the principal targets of the intruders: ls, ps, top, netstat, su, find, ...

This script is executed by check_by_ssh, as the four previous scripts.

With the use of Tripwire, an intruder will not be able to change the monitored binaries. The attacker cannot to hide his presence with modified binaries.

CHKROOTKIT AND NAGIOS

With the popularization of the automated tools of assault, gaining privileged accesses and to conceal them has be-

come an extremely simple task. After the phase of exploration and the phase of obtaining a privileged access, the worry of the intruder centres on the installation of a rootkit that conceals his presence and supports the obtained privileges.

Chkrootkit is a command line tool that detects the presence of rootkits. It uses different methods:

- Checking the promiscuous mode in network interfaces.
- Existence of differences between the processes running in the system according to the command ps and the information of /proc.
- Elimination of entries in the file wtmp, where the login records are stored.
- Checking the opened connections.
- Checking the fingerprints of known rootkits.

Chkrootkit uses some system's binaries for detect rootkits. So Chkrootkit will be trusted if those binaries are trusted. The main group of these binaries are monitored by Tripwire alread. So the responsibility is translated to Tripwire.

What do we monitor?

In the integration of chkrootkit with Nagios a different strategy has been followed that the one used with tripwire. There has been created a script that is executed for snmpd (Simple Network Management Protocol Daemon) and that inserts state information in the tree MIB. This information is gathered by a consultation SNMP. This consultation SNMP is implemented in Nagios using his proper check, check_snmp.

This strategy has been motivated in the long time of execution that uses the test of chkrootkit. The consultation snmpd of Nagios is implemented across a check for consultations SNMP. In this check, it is only necessary to specify the Object Identificator (OID), the machine target, and the name of the community.

The use of Chkrootkit allows to detect the most modern, sophistacated and popular systems of instruders' concealment. Together with Tripwire and the scripts created by authors, Chkrootkit establishes a HIDS capable of recognizing the subtlest signs of instruders' presence.

CONCLUSION

The implantation of a HIDS system formed by several GNU technologies is possible. In the facilities implemented at Ciemat, UAM and UB we monitor to the detail the computing nodes, being capable of detecting the presence of an intruder from his initial steps. This model has proved to be highly effective in the simulated assaults carried out by the authors. Likewise it is of great help for the administrators since the examination periodic and automated with these tools, it allows to save time in the security tasks.

REFERENCES

[1] N. Murillo and K. Steding-Jessen, "Métodos Para Detecçao Local De Rootkits E Módulos De Kernel Maliciosos Em Sistemas Unix", Anais do III Simpósio sobre Segurança em Informática (SSI'2001), (São José dos Campos, SP), pp. 133–139, Outubro de 2001.

[2] "Know Your Enemy: III, They Gain Root", The Honeynet Project, http://www.honeynet.org/papers/enemy3/, March 2000.

[3] "Know Your Enemy: II, Tracking The Blackhat's moves", The Honeynet Project, http://www.honeynet.org/papers/enemy2/, March 2001.

[4] "Know Your Enemy: A Forensic Analysis", The Honeynet Project, http://www.honeynet.org/papers/forensics/, May 2000.

[5] Daniel J. Barrett, Robert G. Byrnes and Richard Silverman, "Linux Secutiry Cookbook", O'Reilly, June 2003.

[6] Reto de Análisis Forense. Rediris. http://www.rediris.es/cert/ped/reto/index.ex.html

[7] E. Pérez Calle, M. Cárdenas Montes, F.J. Rodríguez Calonge, "Using Tripwire to check cluster system integrity", CHEP'04, Interlaken, September 2004.

[8] Tripwire project. http://www.tripwire.org

[9] Tripwire commercial page. http://www.tripwire.com

[10] Chkrootkit. http://www.chkrootkit.org

[11] Nagios monitoring tool. http://www.nagios.org/