USING TRIPWIRE TO CHECK CLUSTER SYSTEM INTEGRITY

E. Pérez Calle[*], M. Cárdenas Montes[†], F. J. Rodríguez Calonge[‡], CIEMAT, Madrid, Spain

Abstract

Expansion of large computing fabrics/clusters throughout the world requires a stricter security to avoid system damages such as data loss, data falsification or misuse.

Perimeter security and intrusion detection system (IDS) are the two main aspects that must be taken into account in order to achieve system security.

The main task of an intrusion detection system is early detection in the previously mentioned cases, as a way to minimize any damage in data held in the system.

Tripwire is one of the most powerful IDSs and is widely used as a security tool by the community of network administrators. Tripwire is oriented to monitor the status of files and directories, being able to detect the lightest change suffered by them.

At CIEMAT, Tripwire has been used to monitor our local clusters, involved in GRID projects such as implementation of LCG prototypes, to guarantee the integrability of data generated, and stored there. It is used as well to monitor any modificacion of operating system files and any other scientific core software.

INTRODUCTION

Security needs for large clusters may be divided into two broad areas:

- Security systems aimed at avoiding unauthorized access to a network (perimeter security).
- Security systems whose target is the detection of unauthorized accesses (intrusion detection systems).

Tripwire [1] is a free software tool included in the second group. It monitors computers' filesystems to discover any modification of the stored directories and files, detecting any unauthorized access as soon as possible. In short, Tripwire's goal is information integrity checking.

GOAL: INTEGRITY OF INFORMATION

A lot of sotfware has been developed to achieve this goal. The aim of this software is to check the integrity of any other software installed on a computer system. The reason to do it is very simple: An attack may be discovered because of modificactions caused to the system. Those modifications are mainly:

System logs: Any access to the system is registered in system logs. System logs monotoring can discover the presence of an intruder, because he would try to hide any entry showing his accesses to the system.

Operating system binaries: If an attacker gains access to the system, he will probably try to hide his presence using Trojan horses. Trojans are software that replace operating system binaries to hide the attacker's processes or files.

Stored information: Misuse of any information stored on the system may cause its alteration, e.g. scientific software.

In short, if a computer program is able to detect any of this modifications, it would be very useful to detect unauthorized accesses. A typical program of this kind of software would work this way:

1. A snapshot of the system is taken before connecting the system to the network. This snapshot is a report about the situacion and characteristics of any file that should be protected.
2. This information may be stored in a safe place, so it cannot be modified (such as a write protected disquette, CD-ROM...) or in a crypted database.
3. System checks are perfomed regularly, and reports are compared with the reference one to detect any modifications.

Following this steps, system integrity checking becomes easier. Any alteration of stored files would make a difference with the snapshot previously taken. Systematic checks would allow to discover any modification to act as soon as posible.

There are a few integrity checkers in the market. *Tripwire* has been chosen because it is the most powerful and versatile and the *de facto* standard nowadays.

WORKING WITH TRIPWIRE

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.

[*] elio.perez@ciemat.es
[†] miguel.cardenas@ciemat.es
[‡] calonge@ciemat.es

- Inode generation number.
- Access Control Lists (ACLs).
- 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 timestamp.
- Inode creation and modificacion timestamp.
- Growing or shrinking files –indicates that the file es expected to grow or shrink.
- Flags.
- Access timestamp.
- Hash checking.

One of the most important characteristics of Tripwire is the use of cryptography. Tripwire protects important stored information by crypting it. It uses two keys: *Site Key*, to crypt configuration files, and *Local Key*, to crypt information about the status of monitored files. Crypted files cannot be modified without knowing these keys, so cryptography is essential to assure system security.

Tripwire is a compound of:

- Policy and configuration files that describe Tripwire's behaviour.

- A database that stores information about the system.

- Reports generated by comparing the original database with the present status of a system.

Policy and Configuration files

Tripwire's configuration file provides general information about the program, such as location of files needed for normal execution (binaries, keys, databases and reports). Some options about how the reports are generated and how using the mail service may be configured here.

The policy file rules Tripwire behaviour. Files that should be monitored are defined here, as well as how this monitorization will be. This file has to be adapted to system configuration so that any unauthorized modification could be detected with enough time.

A free shell script has been developed by Diego Bravo [3] to check policy file syntax. This script adapts a given policy file, deleting any reference to files that do not exist on the system.

```
(
rulename = "Kernel Administration",
severity = $(SIG_HI),
emailto = root@localhost
)
{
/sbin/ctrlaltdel     -> $(SEC_CRIT) ;
/sbin/depmod         -> $(SEC_CRIT) ;
/sbin/insmod         -> $(SEC_CRIT) ;
/sbin/klogd          -> $(SEC_CRIT) ;
/sbin/ldconfig       -> $(SEC_CRIT) ;
/sbin/minilogd       -> $(SEC_CRIT) ;
/sbin/modinfo        -> $(SEC_CRIT) ;
/sbin/pivot_root     -> $(SEC_CRIT) ;
/sbin/sysctl         -> $(SEC_CRIT) ;
}
```

Listing 1: Policy global variables

Any system stores different kind of files who need different levels of protection. Some types of files are defined depending on which properties are monitored. At CIEMAT, files to be checked have been divided into these categories using the following policy:

SEC_CRIT: Files that cannot change. These are system critical files and any modification could mean the administrator has lost control of the system against an attacker.

```
(
  rulename = "Critical boot files",
  severity = $(SIG_HI)
)
{
  /boot    -> $(SEC_CRIT) ;
  /lib/modules   -> $(SEC_CRIT) ;
}
```

Listing 2: Critical files

SEC_SUID: Files with the SUID o SGID bits. These files can be executed by normal user with root privileges. Any file with this property enabled, that has not been installed by the system administration, indicates an unauthorized access.

SEC_BIN: Operating system binary files that should not change.

```
(
  rulename = "Tripwire Binaries",
  severity = $(SIG_HI)
)
{
  $(TWBIN)/siggen     -> $(SEC_BIN) ;
  $(TWBIN)/tripwire     -> $(SEC_BIN) ;
  $(TWBIN)/twadmin      -> $(SEC_BIN) ;
  $(TWBIN)/twprint      -> $(SEC_BIN) ;
}
```

Listing 3: Binary files

SEC_CONFIG: Configuration files belonging to the installed applications. They are seldom changed, but they are often accessed and read by those applications.

```
(
  rulename = "Security Control",
  severity = $(SIG_MED)
)
{
  /etc/passwd    -> $(SEC_CONFIG) ;
  /etc/shadow    -> $(SEC_CONFIG) ;
}
```

Listing 4: Configuration files

SEC_LOG: System logs that store information about any event happened in the system. They grow and become bigger as new information is added.

SEC_INVARIANT: Directories whose access permissions and owner should not change.

```
{
  /        -> $(SEC_INVARIANT);
  /home        -> $(SEC_INVARIANT);
  /tmp         -> $(SEC_INVARIANT);
  /usr         -> $(SEC_INVARIANT);
  /var         -> $(SEC_INVARIANT);
  /var/tmp       -> $(SEC_INVARIANT);
}
```

Listing 5: Invariant directories

In addition to different types of files, three degrees of severity are defined, depending on how a modification would affect system security.

SIG_LOW: Files whose modification would have a low impact on system securtiy.

SIG_MED: Files whose modification would have a bigger impact on system security.

SIG_HI: Critical files whose modification involves a system vulnerability.

Any monitored file would be listed on policy file, followed by the kind of protection and the degree of severity (impact on security).

Tripwire's Database

Once the policy file has been configured and adapted to the system, the Tripwire's database has to be built. Tripwire looks the policy file and analizes every listed file, checking their existence. Its database will be built using this information, it will be crypted and stored. The database will work as a reference to discover any modifications in the future.

Using this original database, Tripwire will check in each execution if the system has been modified comparing present situation with the previous one. Tripwire will generate a detailed report covering any change discovered, as well as the impact those changes would have on system security.

The database is one of the key elements in Tripwire system, therefore is crypted with the local key before being stored. If the database was not protected with cryptography, Tripwire's capacity of detecting any moficication would be seriously damaged, so the computer would be defenceless against an attack.

Tripwire's Reports

Reports are the result of Tripwire's execution. They show the differences found between the original stored database and the present system status. Reports show a general summary where information is divided into several categories, depending on the degree of severity of the violations found.

After the summary, a detailed list of modifications is shown, including added, deleted and modified files. For each file that has been changed, the report shows previous and present status, displaying information such as inode numbers, size, modifty times and checksums.

Reports provide complete information about the system and can be used in forensics analysis.

LARGE-SCALE EXECUTION

Automation and Change notification

Integrity checks are performed automatically at CIEMAT to reduce administration tasks. This can be done adding Tripwire to system's cron. Tripwires allows notifying the administrator any change detected using email.

General information about the mail service (protocol and port used) is specified in the configuration file. Specific information, such as the receiver of sent mail, has to be stipulated in the policy file, as it is shown below.

```
(
rulename = "Kernel Administration",
severity = $(SIG_HI),
emailto = root@localhost
)
{
/sbin/ctrlaltdel      -> $(SEC_CRIT) ;
/sbin/depmod          -> $(SEC_CRIT) ;
/sbin/insmod          -> $(SEC_CRIT) ;
/sbin/klogd           -> $(SEC_CRIT) ;
/sbin/ldconfig        -> $(SEC_CRIT) ;
/sbin/minilogd        -> $(SEC_CRIT) ;
/sbin/modinfo         -> $(SEC_CRIT) ;
/sbin/pivot_root      -> $(SEC_CRIT) ;
/sbin/sysctl          -> $(SEC_CRIT) ;
}
```

Listing 6: Chage notification by email

The policy file allows the administrator to specify which notifications should be sent by mail, indicating the mail address of the receiver (e.g. email notifications can be restricted to modifications affecting critical files).

Centralization

Tripwire can be used more effectively if its execution is centralized. This can be done defining a shell script in a central computer. This scripts run routinely and use Secure Shell (SSH) to throw parallel executions of Tripwire on those systems that should be monitored. After all executions have concluded, the script picks up all the generated reports. A whole network can be monitored from a central computer working this way.

Centralized execution using SSH has been tested at CIEMAT, but our approach to this goal implies mainly the use of Simple Network Management Protocol (SNMP). A declaration of state may be inserted in the local MIB tree after Tripwire's execution. Then, a client can pick up all this information from the server nodes in the cluster using SNMP command *snmpget*. This task may be easier as programming a shell script as the communication protocol is already implemented.

SNMP requires the installation of a daemon in the cluster, to fill the MIB tree with information about monitored properties, and a client, to get this information, in a central computer. At CIEMAT, about 35 computers are currently monitored in our local clusters from an unique machine. The system load has been monitored in this machine, and results are relatively low, since Tripwire centralized execution needs very little time of CPU.

About 45 machines are being monitored in other collaborating institutes, such as UAM and UB in a very similar way, following the protoype implemented at CIEMAT.

User Interface

Centralized execution would be easier to do if Tripwire had an user interface, but there is none licensed under GPL [4] at the moment.

Anyway there are several options to use an interface, such as integrate Tripwire in a Monitoring System. At CIEMAT, and in other institutes, we have chosen Nagios [5], a security toolkit widely used for monitoring, as an user interface for Tripwire.

Nagios allows the execution of remote security checks through SSH or SNMP (using *check snmpparameter*) and has been used [6] to provide a graphical interface showing the results of the checks performed, making easier the monitoring of large clusters.

ALTERNATIVES TO TRIPWIRE

A well known alternative to Tripwire is AIDE [7], which is licensed under GPL.

AIDE is also an integrity checker that looks for differences between a filesystem and a previously built database, but it does not use cryptography to protect it. In fact, AIDE does not use a database stored on the monitored computer. It works comparing local files to reliable ones stored on a different filesystem. Reliable files have to be written on a local CD-ROM for each monitored computer or be mounted remotely. That is an additional difficulty.

Tripwire's database may also be stored in a non writable filesystem, such as write protected disquette or a CD reader. But as the database is crypted, an attacker will not be able to modify it to hide any changes done to the system as a consequence of his attack, and any unauthorized access will be detected. Tripwire does not need to involve external facilities to guarantee its security, and this is an advantage over AIDE.

CONCLUSIONS

Security checks are essential in any computing facility. Tripwire is a very comprehensive tool that allows to monitor each computer in a network defining in a very precise way the policy to follow.

The possibility of a large-scale centralized execution of Tripwire and the possibility of adapting it to the needs of a given cluster, simplifies management without limiting facilities and makes possible the monitorig of large and complex cluster systems such as LCG. [8]

Finally, the inclusion of Tripwire in the Nagios monitoring toolkit provides a very versatile and secure environment for integrity checking that has been largely tested at CIEMAT, and other collaborating institutes.

ACKNOWLEDGEMENTS

Authors wish to thank Dr. José M. Hernández for useful comments.

REFERENCES

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

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

[3] D. Bravo Estrada, “Guía breve de Tripwire”.
http://es.tldp.org/Tutoriales/GUIA_TRIPWIRE

[4] General Public License (GPL).
http://www.gnu.org/copyleft/gpl.html

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

[6] M. Cárdenas Montes, E. Pérez Calle, F.J. Rodríguez Calonge, “Using Nagios for intrusion detection”, CHEP'04, Interlaken, September 2004.

[7] AIDE project.
http://sourceforge.net/projects/aide

[8] LHC Computing Grid project (LCG).
http://lcg.web.cern.ch/LCG/