

The Trouble With Tripwire: Making a Valuable Security Tool More Efficient

Edward R. Arnold 2001-06-06

The Trouble With Tripwire: Making a Valuable Security Tool More Efficient

by *Edward R. Arnold*

last updated June 6, 2001

Background

Even with the protection that a security perimeter may provide, the fact remains that firewalls aren't foolproof and that potential attackers are hard at work, 24 hours a day. No networks are entirely secure. Furthermore, some organizations must allow remote logins to machines from sites outside their perimeter, meaning they must maintain a certain number of semi-exposed hosts that are vulnerable to attack. It is not enough to know that a system has been scanned or probed. If an attack is detected, how can it be determined that a system has actually been compromised and important files removed or altered? One answer is to always run [Tripwire](#), or one of its free-ware cousins, in order to detect changes in critical system files.

Unfortunately, this is not always an efficient solution. The author, who works for a well-known research institution in the USA, has experimented with Tripwire since the mid-nineties. The institution had experienced a series of hacks, as well as a number of mishaps stemming from incorrect machine configuration changes that led to internal, and external, embarrassment. These problems led them to try Tripwire to detect if any important files on their systems had been changed. Unfortunately, due to the volume and frequency of reports that it generated, it quickly became clear that Tripwire was a labor time-sink!

Given that running Tripwire may be a security necessity, it would be beneficial to do so as efficiently as possible. This paper will focus on ways to reduce the time and labor required to effectively operate run the Tripwire security and configuration monitoring tool.

Tripwire: Boon or Bust?

Tripwire is much like the fabled elephant and the blind men: how you feel about it depends on the perspective from which you approach it. A person who has successfully used Tripwire to detect cracked binaries and/or system misconfigurations will have nothing but praise for it. On the other hand, someone who has been "stuck in the trenches" reading through endless reports

in an attempt to find problems, will think that it's a labor-intensive waste of time. Minimizing the labor required dictates that reports be as brief, and as infrequent, as they possibly can be made. Using Tripwire on a day-to-day basis can be an uncreative and essentially boring activity. On the other hand, if one can reduce the torrent of data that Tripwire provides, and thus make it simpler to use, then running it can become bearable (if not necessarily palatable.) Fortunately, it is possible to reduce the time and effort required to administer Tripwire, primarily by reducing the time required to process reports.

The Time Sinks

More than a year of experience in running Tripwire has made it pretty clear where the inefficiencies within the product lie, as of commercial version 2.2.1 (Linux) or 2.2.1A (Solaris). The following are some of the areas that the author believes contributes to time and labor inefficiency of Tripwire.

History Mechanism

The single most important time efficiency issue with Tripwire is the lack of a report history mechanism, which would drastically reduce the number of reports. For instance, a dozen systems being checked three times per day can result in over 1000 reports per month, any one of which could contain the critical information the tool is supposed to detect. Even the most careful tuning cannot prevent this; for instance, the installation or modification of a large software package may suddenly result in a large report that will continue until the administrator has time to do a database update. This is especially true if an administrator cannot make policy tuning and database update his first priority.

Policy Tuning

Tripwire does not have the capability to automatically tailor rules in the policy file (to minimize report volume) by watching a system over a period of time in order to recommend rule changes. At install time, the distributed policy file per processor/OS type is just a static file that comes with the distribution. The install procedure does not customize the file based on simple criteria, such as the presence of certain files or filesystem structures revealed by the mount table. Furthermore, it excludes some important files (e.g. the root crontab). Since the same template policy file per processor/OS type results upon install, a lengthy tuning process is required with each machine in order to minimize useless report output. Currently, there is no

way around this problem other than to try to make the tuning process as quick as possible until most noise is tuned out. In the meantime, a history mechanism would significantly lessen the urgency of tuning.

Report Formats

Although the commercial Tripwire product has five report formats, none of them offers a maximally-abbreviated single-line format that provides violation type, filename, and changed attribute keys in a single line. (Tripwire's closest format is level 1, which provides only violation type and filename.) The default (level 3) format is a multiple-line format that is time-consuming to use. In many cases, the signature associated with a changed attribute isn't even of interest (for example, who wants to stare at an MD5 signature?) and, in many cases, the signature associated with a file attribute could change frequently and defeat history. So it is ultimately up to the system administrator to inspect the attribute or file in question before doing database or policy update. We parse the default Tripwire format into a single-line-per-file, 3-field format which looks like:

```
a /etc/inet/core  
m /etc/passwd smCM  
r /local/etc/named.foobar
```

Although file attribute values are currently not included in the report, file content signatures may be added at a later time if automated diffs are implemented (see "Automated File Diffs" below).

Ease of Maintenance

Tripwire database and policy file maintenance are made easier if the Tripwire admin does not have to remember argument switches and long filenames. Maintenance time is also a good time to enforce security issues that the harried or newbie administrator might forget, such as making sure Tripwire passwords are protected by doing an update on the system console or via a secure shell link.

Lack of Regular Expressions

The Tripwire policy file allows complete exclusion or lower security policies on directory trees.

However, it does not allow exclusion or changed security policies on regular expressions for filenames. Directories in which pseudo-random filenames are created by utilities that are not always under the Tripwire administrator's control - other than places like /tmp where they are expected - create "noise" in report output.

Another aspect of the regular expression problems is that it is not possible to instruct Tripwire to report on the addition of filenames that have historically been associated with cracking activity, for example: `"/tmp/\..\.*"`.

E-Mail Report Minimization

Tripwire now allows e-mail reporting to go to different addresses for different portions of a machine's filesystems. Unfortunately, smaller organizations don't have the staff to assign responsibility for a system to several people. The best approach for such small organizations is to have no more than one report per Tripwire check run, and to have one primary person responsible per system.

Reducing the Time Sinks of Tripwire

Despite the numerous shortcomings listed above, it is possible reduce the labor and anxiety required to use the base Tripwire product, Tripwire for Servers, by taking the following steps:

- Implement two wrappers, one for routine checking, and one for database or policy update;
- Enforce security in the database and policy update wrapper, e.g. require ssh or console use, and provide reminders about ro/rw disk use. Allow checking to be done as a side-effect of database update so that a new report baseline can be established in a single step;
- Provide labor-saving convenience features in the update wrapper. For instance, policy update (which is CPU intensive) can be easily re-entered so previous changes aren't lost if the administrator has to terminate prematurely;
- In the checking wrapper, parse the default report format into a single-line-per-file compact format at time of checking;
- In the checking wrapper, implement a simple history mechanism and file exclusions to reduce duplicate reporting. A simple (diff-based) mechanism proved to be as good as an earlier, more complex mechanism that considered more than two previous reports. We

have chosen to disregard the complexity that would be introduced by using the capability to specify severity level or rulename at check time, which we haven't found useful in our environment.

Further Improvements

While the simple steps discussed so far will reduce the labor required to use Tripwire, a further reduction is possible within the structure/limitations imposed by program's architecture. It is also apparent that further gains in efficiency are necessary before Tripwire is quick and easy enough to use that people will not mind using it. In order to further improve the efficiency of operating Tripwire, the author suggests that the developers incorporate the following improvements into the next commercial version.

Abbreviated Reporting - On machines that are used for software development, an update of a software package (located in an area subject to Tripwire monitoring) that resides under its own root location can result in a large e-mail report. In such a case, very little information is provided by listing every changed file. This can be dealt with by generalizing the exclusions file to a check-time configuration file that contains exclusions and abbreviations. Abbreviations are filename wild-cards that cause a head directory name to be reported just once with a count of the number of files affected, if abbreviation mode is turned on at check time.

Read-Only Disk on Redhat Linux Hosts - Use of a switchable read-write/read-only SCSI disk requires that a SCSI reset be done when the protection switch is changed. Part of the author's development effort included writing a "remount" command that uses the SCSI reset capability available in user-level as of Solaris-7. (This feature had previously been used on IRIX-6 machines; IRIX-6 provides a canned user-level command for SCSI reset.) The author is currently investigating to what extent a ro/rw disk can function under Linux.

Speedier Database Update - Database update and re-check requires excessive elapsed time because the default update method requires an interactive editing step. Developers should investigate approaches that could place all of this activity in the background without unacceptable security risk, e.g. without Tripwire passwords visible in the process table.

Automated File Diffs - A change to an ASCII text file, e.g. "/etc/passwd", creates an obligation on the part of the Tripwire administrator to determine if changes are benign or not. The obvious way to do this is to diff the file against a copy of the same file taken from a backup

prior to the file's last mod date. Unfortunately, this procedure is very time-consuming and so administrators tend to skip it except in unusual cases. Since some sites have automated backup systems from which files can be retrieved by name, Tripwire Inc. should look at development of an API for interfacing with these systems. If this proves to be feasible, it would probably be implemented only for files specified in the check-time configuration file, and file content signatures could be added to the abbreviated report format to support this.

Edward R. Arnold works for a well-known research institution in the USA. He has presented two papers at USENIX conferences on the topics of automated backup and configuration control. Those who desire further detail or code may contact Mr. Arnold at era@pobox.com.

[Privacy Statement](#)

Copyright 2006, SecurityFocus