

Securing Linux with AIDE

Kristy Westphal 2001-05-09

Securing Linux with AIDE

by *Kristy Westphal*

last updated May 9, 2001

It is often difficult to assess damage from an intrusion through syslogs and good old-fashioned sysadmin work alone. The good news is that there are tools available to help assist in this battle, tools known as host-based intrusion detection systems. There are numerous versions of this type of software (Tripwire, Tiger), but this article is going to discuss one in particular that can be used to help secure various types of UNIX operating systems - AIDE. This article will specifically discuss the ways that AIDE can be used to protect a Linux workstation.

AIDE (Advanced Intrusion Detection Environment) can be used to help track file integrity by comparing a 'snapshot' of the system's files prior to and after a suspected incident. It is a freeware version of Tripwire, created and maintained by Rami Lehti and Pablo Virolainen. Essentially, AIDE uses a database to accumulate key file attributes like permissions, mtime, ctime, and number of links for a system. The idea is to build the database before 2 things occur:

1. before the snapshot image of these files is taken prior to the system being placed on a network; and,
2. before the snapshot image is taken prior to a system compromise.

For further protection, a checksum of each file in its current state can be used with a choice of several hash methods.

The idea behind AIDE and other host-based IDSs is for the snapshot to be taken and then periodically updated as the system is updated. Patches, hardware and other software installs tend to change the size and nature of files, so it is always a good idea to re-run AIDE after making changes. If the administrator then suspects that a system has been compromised, running AIDE and then comparing the two snapshots will assist the admin in honing in on what happened. Doing so without this original snapshot can be difficult at best.

How to set up AIDE: Key Ingredients

There are several key ingredients that are needed to build an AIDE installation (per the .1 version of the AIDE Manual):

- GNU Make
- Gcc or similar ansi C compiler
- GNU Flex
- GNU Bison
- Mhash library
- The postgres sql developer library (if postgres is to be used as the back-end database)
- The source code for AIDE

For this article, the author will be using RedHat 6.2 (fully patched and recommended security fixes in place) [see Relevant Links section at the end of this article] on an Intel Pentium processor. Although Linux comes with many of the above components natively (gcc, make, postgres), it is recommended to make sure that all the latest versions are used. Accordingly, the author used gcc 2.95.3, make 3.79, flex 2.5.4a, bison 1.28, mhash 0.8.9 and postgres 7.0.3. Of course, that doesn't always mean smooth sailing while building software, so it is also recommended to keep the tar files of old versions handy in case problems are encountered. It is also advisable to install the packages in this order to ensure that each has what it needs to be installed correctly.

Architecting your setup

After updating the versions of the base ingredients, the user arrives at a critical juncture in the installation: how should the application and database be set up to ensure the maximum security of its contents? The application itself is probably fine on the server - given that the permissions to it are set correctly. It is recommended to set up an application account to own it, with an application group to go along with it. An account along the lines of 'aide' with a group of 'aide' should work. Then establish permissions as follows:

```
chown -R aide:aide /aidedir
chmod -R 700 /aidedir
chmod -R 700 /usr/local/bin/aide
chown -R aide:aide /usr/local/bin/aide
```

Where /aidedir is where you have installed the application. This way, whenever using AIDE, users will log in to the aide account and run it using this account. Also, accountability of what actions are undertaken by the application owner is increased as opposed to trying to track everything from root. Further, the application itself can be easily restored if compromise is suspected. Lastly, when AIDE is run for the first time, users must make sure that the /aidedir is in the config file, so that they have an original snapshot of what they have installed.

That leaves the database portion, as well as the config file that establishes what files are being stored in the database. Several methods of protecting this data can be undertaken. First, the database and

config file (after setup and the initial data capture), can be written to a disk that can be unmounted and kept offline until it is needed. On networks with multiple servers, a good enterprise solution is an automated process by means of a cronned script that does the following:

1. pulls the database and config files from each server you have it installed on;
2. store them all on one disk, segregated by which server it is from (i.e. a hierarchical directory structure with the top level directory being the name of the server);
3. run an MD5 checksum against each separate file, storing the information in a separate log file (i.e. `md5 aide.db >> /tmp/aide_md5_print`);
4. unmount the drive and remove it from the system;
5. put it in an anti-static storage bag;
6. then either store it in that fireproof lock box or offsite somewhere, using it only when you need to run your quarterly checks and updates.

There are other options, of course. If a disk is not available, then other types of removable media like CD-Rom or tape can be used. CD-Rom is preferable to tape because it is more stable and is (for the most part) write once, read many. The same principle of backing up all of servers to one central location, then writing them to either tape or CD-Rom applies. Depending on the environment, either of these options will work, so users should plan to protect this data accordingly.

Things to watch out for

The best practice for installing this type of intrusion detection system is to begin with a system that has no operating system installed, is not accessible on the network, and is installed by the user (and only the user) to ensure that the system is installed properly. Installing a host-based IDS is best done in an environment that is clean, where it can be certain that no tampering has been done, and good log of system installation and maintenance has been kept.

In reality, an administrator's systems are already online before they read or learn about something like AIDE. In that perfect world, it would be wonderful to have AIDE installed and working prior to placing the system on the network. So, what can be done with those systems already on the network? Users can run AIDE anyway, knowing full well that the data and operating system could possibly already be compromised. They can take a look at the data produced from AIDE after they run it for the first time to see if they can pick up any anomalies right away. Sometimes you may spot something that doesn't seem quite right and then you can investigate further. Better yet, find a system that is brand new and not on the network and compare your existing system AIDE report to an AIDE session run on the new system. Then wait a couple of weeks and run it again to check for anomalies and compare it to other systems. Also ask yourself if you have noticed anything odd about these systems.

If your firewall reports odd traffic to an unknown host every time you run 'ls' on a certain server, there may be bigger problems here.

One warning about the install is that gcc takes a very long time to upgrade - gcc is a big package and takes a while to compile. On another note, use `./configure --with-mhash` to enable the additional hash functionality when you are configuring the AIDE software. Otherwise, the installation is typically very smooth.

How to set up the Database: How Rules are Matched

As with any type of intrusion detection system, rules files always pose a challenge. Weeding through what the user wants to use and not can take a lot of patience and analysis. However, it is not an impossible task! The AIDE config file can be very detailed in what it examines, depending on the security policy of the site or organization in question. The rigorous site that likes to keep a record of everything may want to have a full blown config file with all checks turned on, and in turn needs a sizable disk for storing the database. On the other hand, the 'I just want to make sure nothing has been changed' site might want to think about using a set of files commonly altered during hacks and check a few characteristics of these files for issues. It all depends on what the system administrator ultimately wants.

Somewhere in the middle ground, sysadmins may find that a combination of all characteristics on key files and only a few on other files may prove most useful. This requires a bit of effort on the administrator's part. A good place to start this approach is to first understand how rules are matched. For a very detailed explanation of the features in the config file, see the `aide.conf` man page that gets neatly installed during the AIDE compilation.

First off, there are three types of lines in the config file: configuration, selection and macro lines. The configuration lines set the parameters and variables, while selection lines display what files will be captured in the database. Lastly, the macro lines define or undefine variables within the config file itself. Note that entries in the `aide.conf` file are case sensitive and that white space is ignored.

When the config file is processed, AIDE separates out rules into several regexps nodes within a tree: equals, select and negative select. The regexp that is the first character in the rule determines which node to place the rule in. AIDE will also add an implicit '^' in front of each rule. So, when AIDE runs through the rules to find a match for a file, it looks to the equals node first, then the select list, then the negatives list, then makes a decision whether to add it to the database or not.

A brief explanation of regexps

What are regexps anyway? Regexps is short for regular expressions, and is used in Perl to do character-based mapping. Regexps can be interpreted slightly differently depending on what operating system you use. Generally, some characters are special constructs and the rest are ordinary. Ordinary characters are those that are interpreted just as they look. But special characters imply something else. For instance, a '\$' means it looks for a match at the end of a line, a '^' matches an empty string, but only at the beginning of line, and a '\' means two things: 1) it quotes special characters and 2) it introduces special constructs. ("[Syntax of Regular Expressions](#)", Section 10.5 of the GNU Emacs Manual, March 29, 2001

When using these special characters in the aide.conf file, a number of odd things can happen if the users try to match rules without keeping in mind the order in which the rules are processed. For instance, a user may want to check the attributes for all files under /etc, except for the ld.so.conf file, in which case he or she might write a rule to look for:

```
/ R (permissions, inode, number of links, user, group, size, mtime,
ctime,
and use the md5 checksum hash to store the files)
!/etc/*.conf
=/etc$ R+a (all of the above plus atime)
```

This would mean that the user negates all of the conf files, but then turns around and adds them all back in, which is probably not something that users want to do. By being more specific by changing the second line to read

```
!/etc/ld.so.conf
```

and flipping the order of the second and third lines, users will accomplish what they are after.

The lesson here is that users must not be lazy and must remember to be specific when using regular expressions. They must remember also to always test their rules and examine their log files in order to ensure that the rules that are established are logging what the user intends.

Troubleshooting your configuration

There are a few tools that can be used if AIDE does not seem to be functioning properly. First, users can run AIDE in verbose mode to grab debugging information. This can be done with the '-verbose=225' variable. Second, users might want to make sure that their regexps knowledge is sufficient, and read up on the topic to make sure their rules are set up properly.

Using AIDE

When first using the AIDE binary, users should use the '--init' variable to create the database, which is a file called aide.db.new. They will then want to make a copy of this file, calling the copy aide.db. When users are ready to run AIDE to check for anomalies, they will want to use the '--check' variable, and dump the output to a file for review. They can run something like this:

```
aide --check > /tmp/aide.log
```

Finally, once changes have been made to the system and users want to update the current database, they can make an archive copy of the aide.db.new file (copy it to aide.db.new.1), then run aide with the '--update' variable, and copy the new aide.db.new file to aide.db.

When to use and not use AIDE

AIDE will not solve everything - this is true of most intrusion detection systems; however, it can certainly help shorten the investigation time during incident response by focusing in on the files that have been changed. It may also be useful if the system administrator, has implemented changes that did not go as planned. By running AIDE prior to any updates, the sysadmin can get a quick idea of what may have gone wrong. The uses are numerous.

The implementation scheme of AIDE again depends on the security philosophy of the site. If the policy is to protect only servers located in your DMZ, this is possible. If an organization has an engineering staff with root privilege who like to play with their systems rather than do their work, then it might be good to install AIDE there. If the sysadmin likes consistency across all servers and workstations, it is possible to write a nice script that pulls down all the required packages, installs them in the proper order and runs AIDE. It can then make a copy of the database file, and then set up a cron job to run the same process weekly, mailing the administrator a differential of the new database and the old to look for changes.

Reviewing your logs

Hopefully, once the aide.conf file has been set up properly, the administrator won't often have much to review. There aren't really any log files produced by AIDE, so the user simply has to make a backup copy of the old aide.db file for reference and redirect output of the "aide --check" command to a file. To be of any use, this must be done regularly! It should be put on the list of logs to review and done at whatever interval the admin chooses to run it at (weekly seems to work nicely, unless there is reason to think it should be run more frequently, i.e. servers in your DMZ that run services like anonymous FTP and web sites).

Otherwise, if the administrator simply wants to keep these files for record keeping and for the

occasional incident handling situation, then he or she can run an md5 checksum on the aide.db file, store this and the database file itself at another location or read-only media for safekeeping.

Methods for cleaning up ruleset

As with any intrusion detection system, host- or network-based, it can take some trial and error to get the rules or config files running just the right way. This takes some patience and time, which is usually in short supply. One method that can be used to help weed through this process is fairly straightforward: first, the user should enable everything and then read the reports and look for things that are either redundant or don't appear to be important. The user can then eliminate unnecessary files until he or she is happy.

Another approach is to look at the overall picture of your security methodology at the OS level. Do you like to know everything that is going on at all times with your system for logging purposes? Perhaps it is a business requirement that you track changes in certain data files or retain logs on certain files for a certain period of time? Or do you only care about certain directories that would be targets for hackers? Are you short on storage space and need to concentrate on key files only? Once you can answer these questions it becomes clear very quickly what your config file might look like.

One last piece of advice on the ruleset: directories that change frequently should not be included in your config file. Directories like /tmp, /mnt and /proc are often not good directories to track because it is their nature to change often. Remember, host-based intrusion is focused on files that shouldn't change often but, for some mysterious reason, do.

Conclusions and thoughts on host-based IDS

If you have ever been in the situation where you suspect that a host has been compromised but have been unable to determine what might have been changed, a host-based intrusion detection system like AIDE can be very helpful. Host-based IDS's help you to focus in on just what has changed, rather than chasing down guesses and wasting valuable time.

AIDE is a valuable, configurable tool that can take a snapshot of a system in its original state in order to track variances in the system. The list of benefits that a host-based IDS like AIDE offers an overworked administrator is seemingly endless. These benefits can include tracking of binary and file integrity as well as tracking of system directory integrity. Further, AIDE and other host-based IDS's can assist sysadmins in change management. For instance, it can track changes made by other system users who neglect to notify other team members of the change, resulting in a system that won't boot up at all.

On the downside, host-based IDSs can be a bear to install across an enterprise. Without automated tools to help setup and distribute the software to each system, it can be time-consuming to get the IDS installed. It also takes time to tune and set the conf file in the optimal manner. It is important that administrators use their best judgment when setting this type of system up in their environment. It is advisable that they weigh the benefits versus the risks and implement the system accordingly.

Kristy Westphal is a versatile network administrator, skilled in troubleshooting and process analysis. She is knowledgeable in several flavors of UNIX and NT, as well as various aspects of information security and disaster recovery planning. She is currently employed by KPMG LLP in their Information Risk Management practice as a consultant.

Relevant Links

[Securing Linux](#)

Dale Coddington, SecurityFocus

[Securing Linux, part 2](#)

Dale Coddington, SecurityFocus

[Securing and Optimizing Linux, RedHat Edition](#)

Gerard Mourani

[Privacy Statement](#)

Copyright 2006, SecurityFocus