

Sunday, August 27, 2006

Analyzing aide (advanced intrusion detection environment) output with PHP

Since we started hosting our sites on our own server we had some nasty cracker-attacks (most often certainly script-kiddies) causing lots of traffic by hosting crappy italian movies or by installing rootkits. To at least have a chance recognizing whether the system had been compromised we started to use aide some time ago. Aide keeps track of changes in the filesystem and provides us with a human-readable report once a day.

However, sometimes changes in the filesystem happen because of (security-)updates and not because a cracker exchanged your ps binary against his own personal version. Of course aide has no chance to identify such valid and invalid changes. 3rdPEARty's Util_AideAnalyzer is a solution to that problem - read on if you're interested.

The output aide produces when it checks the filesystem against the database looks something like this:

AIDE found differences between database and filesystem!!

Start timestamp: 2006-08-27 12:48:40

Summary:

Total number of files: 78400

Added files: 44

Removed files: 7

Changed files: 958

Added files:

...
added: /var/backups/dpkg.status.2.gz
added: /var/backups/dpkg.status.3.gz
...

Removed files:

...
removed: /var/backups/dpkg.status.4.gz
removed: /var/backups/dpkg.status.5.gz
...

Changed files:

...
changed: /usr/lib/menu
changed: /usr/lib/menu/lynx
...

Detailed information about changes:

...
File: /usr/lib/menu/lynx

Blog Export: a programmer's best friend, <http://blog.php-tools.net/>

```
Mtime : 2005-10-08 13:30:57 , 2006-08-14 01:36:12
Ctime : 2006-06-17 16:14:23 , 2006-08-26 16:19:35
Inode : 5947970 , 5948063
```

...

It lists some statistics, a list of added, removed and changed files/directories and a detailed summary of what exactly changed.

As mentioned above the summary often contains files/directories that were changed because of updates on our machine. To avoid the tedious task of separating the good ones from the bad ones you can put the output into a file and let Util_AideAnalyzer do the filtering for you. The following code snippet shows you how easy it is.

```
// necessary includes
require_once '3p/Util/AideAnalyzer.php';
require_once '3p/Util/AideAnalyzer/ItemChecker/Ctime.php';
require_once '3p/Util/AideAnalyzer/ItemChecker/Mtime.php';
require_once '3p/Util/AideAnalyzer/ItemChecker/Composite.php';

// the file containing the results of the aide-check
$file = 'rsrc/check-2006-08-27.txt';

// Analyzed-item checkers to be used in compound
// to sort out valid changes.
$checkers = array(
    new Util_AideAnalyzer_ItemChecker_Ctime(strtotime('2006-08-27 12:43:26'), 10),
    new Util_AideAnalyzer_ItemChecker_Mtime(strtotime('2006-08-27 12:43:26'), 10),
    new Util_AideAnalyzer_ItemChecker_Ctime(strtotime('2006-08-26 00:50:55'), 10),
    new Util_AideAnalyzer_ItemChecker_Mtime(strtotime('2006-08-26 00:50:55'), 10),
    new Util_AideAnalyzer_ItemChecker_Ctime(strtotime('2006-08-26 16:19:33'), 10),
    new Util_AideAnalyzer_ItemChecker_Mtime(strtotime('2006-08-26 16:19:33'), 10));

// a composite analyzed-item checker
$compChkr = new Util_AideAnalyzer_ItemChecker_Composite($checkers);

// creating an analyzer instance and performing analyzation
$analyzer = new Util_AideAnalyzer($file, $compChkr, false);
$analyzer->analyze();

// print the results
echo $analyzer->printTestResult() . "\n\n";
echo $analyzer->printDebugInformation();
```

You see that I create six item-checker objects (for each date one mtime and one ctime checker) which will sort out items whose mtime/ctime property changed at the given time (with a bias of 10 seconds, which is the second parameter). Each of those three timestamps represents a date/time when I updated the system. Thus the following change aide noticed:

```
File: /usr/lib/menu/lynx
Mtime : 2005-10-08 13:30:57 , 2006-08-14 01:36:12
Ctime : 2006-06-17 16:14:23 , 2006-08-26 16:19:35
Inode : 5947970 , 5948063
```

is a valid one, now (compare the ctime-property and keep the 10 second bias in mind).

The rest is as simple as cooking coffee. I take the \$checkers array and create a composite checker implementing the Util_AideAnalyzer_AnalyzedItemChecker interface. Then I create an Util_AideAnalyzer instance and call the analyze() method. The rest is just printing out the filtered information which will look like that:

Invalid items:

...
File: /etc/postfix/prng_exch
Directory: /root
File: /root/.viminfo
File: /root/.bash_history
File: /root/.rnd
File: /home/luckec/.bash_history
...

Valid items:

Directory: /etc
Directory: /etc/alternatives
...

Analyzation statistics:

Number of added items : 44
Number of removed items : 7
Number of changed items : 958

Number of valid items : 904
Number of invalid items : 54

Number of added/removed/changed items: 1009
Number of valid/invalid items : 958

I think you will agree that checking the validity of the remaining changed files/directories (invalid items) is done pretty fast as there are only a few left.

If you like what you read feel free to download and install Util_AideAnalyzer. Type

```
pear channel-discover 3rdparty.net
```

Then you will be able to install the package by using:

```
pear install 3p/Util_AideAnalyzer
```

Have phun!

Posted by luckec in PHP at 22:47

Sunday, August 20, 2006

New releases of patTemplate and patForms

After I finally finished my book (and my contribution to Exploring PHP) I finally have enough time to work on all of my open source projects again and today made two new releases. patTemplate 3.1.0b1 contains tons of changes since the last stable version. Most of the features that users have been requesting for years have been incorporated into this release, like the possibility to read templates from a database: \$tmpl = new patTemplate();

```
$tmpl->setRoot('mysql://root:@localhost/test', 'DB');
```

```
$tmpl->readTemplatesFromInput("SELECT content FROM templates WHERE id='foo'", 'DB');
```

// or use this syntax:

```
$tmpl->readTemplatesFromInput('templates[@id=foo]/@content', 'DB');
```

It is now also possible to use any PHP function or method as a default value for a variable which allows you to prefill a value with the current timestamp. Furthermore I have been putting a lot of work into the new patTemplate manual.

patForms 0.9.0b3 mostly contains bugfixes and provides several install-groups for the PEAR installer, which allow you to choose your form renderer when installing patForms. This will hopefully be one of the last beta version for patForms 0.9.0 as we plan to release a stable version during the next 4-6 weeks.

Both packages can easily be installed using of PEAR channel server at pear.php-tools.net.

Posted by schst in PHP, PAT at 15:25

Creating PEAR-installable nightly builds

If you incorporate a "release early, release often" policy for your projects, it helps you to detect bugs in an early development stage of your application. In many cases making a release takes some time and you do not release new versions as often as you intended. We had these problems with our projects over at www.php-tools.net. That's why we created snaps.php-tools.net, a site where you can download nightly builds of our projects as ZIP, TAR.GZ or TAR.BZ2 archive. This saves the users, who want to test the latest development versions the hassle checking out the latest version from our subversion repository. The problem of this technique was, that most users use the PEAR installer to deploy our packages and if they downloaded a ZIP file, it's not possible to use the PEAR installer on this file. That's why we came up with a solution to create nightly builds that are installable with the PEAR installer. Read on, if you are interested in how this can be achieved...

To create a PEAR installable package, all you need is a package.xml file, that contains all information about the project and the related files. So to create a nightly build, you will need a current package.xml that describes the package. Creating this by hand can be very cumbersome, as it has to contain all dependencies as well as all files that you want to include in the resulting package. But help has arrived, as Greg Beaver provides a PEAR package that aids you in the creation of package.xml files: PEAR_PackageFileManager. This package provides a PHP-API to read, modify and write the package.xml file needed to create an installable package. To automate this process, you need to follow these steps: Export the latest version from Subversion/CVSDetermine a version number to identify the nightly buildGenerate a package.xml fileUse the pear package command to create the archiveThe first task at hand is extremely easy, as Subversion and CVS both offer the export command. To get the latest version of patTemplate, the following command does the trick:

```
$ svn export http://www.php-tools.net/svn/patTemplate/trunk patTemplate
```

Now that you have exported the latest code-base the next steps is to create a unique version number. We decided that we wanted to use the latest officially released version (e.g. 3.1.0) and add a version suffix. This suffix could either be the latest SVN revision, or the date that the last change has been made.

As the revision in Subversion is incremented for the complete repository, once a commit has been made and all our projects share a common repository, we decided to use the date of the last change in the project folder as the version suffix. Once you have checked out the contents, this can be easily accomplished using the SPL's

```
RecursiveDirectoryIterator:function getLastModifiedDate($folder) {  
    $modified = 0;  
    $dir = new RecursiveDirectoryIterator($folder, RecursiveIteratorIterator::LEAVES_ONLY);  
  
    foreach (new RecursiveIteratorIterator($dir) as $file) {  
        if ($file->getMTime() > $modified) {
```

Blog Export: a programmer's best friend, <http://blog.php-tools.net/>

```
        $modified = $file->getMTime();
    }
}
return $modified;
}
```

This information now has to be used in a script, that will generate the package.xml file. For snaps.php-tools.net, we decided to split these tasks. The SVN export and the generation of the version suffix is handled by a simple script `createSnaps.php` which also build the plain ZIP or TAR archives.

After exporting the latest SVN version, this script searches for a file called `autopackage2.php`. If the file is available, it will execute the file and pass the version suffix as the sole parameter. `$createPackageFile = "php autopackage2.php " . date('YmdHi', $mtime);`
`exec($createPackageFile);`The `autopackage2.php` file then will use `PEAR_PackageFileManager` to create the current package.xml file. I will not go into detail here, you may take a look at an example in the `patTemplate` subversion repository. The most important part is, how the version number is created:// Base version
`$baseVersion = '3.1.0';`

```
// current version
$version = $baseVersion . 'dev' . $argv[1];
```

Furthermore it uses a `filelistgenerator` provided by `PEAR_PackageFileManager` to include all files that have been exported from SVN. This `autopackage2.php` script will generate a `package.xml` file and save it on disk. To create a PEAR installable package from this newly generated `package.xml` file, all that's left to do is execute the following command:`$ pear package -n`The `-n` switch is used to echo the full path to the generated `.tgz` package file after it has been generated. We use it to move this file to the document root, so it can be downloaded. And voilà you created a PEAR-installable nightly build.

For snaps.php-tools.net we also implemented some more features:It is possible to not only create a version from the current trunk, but as many branches as you likeWe also create ZIP, TAR.GZ and TAR.BZ2 archives for users who do not use the PEAR installerWe implemented a simple class that acts as a webfrontend to download the nightly builds.If you are interested in using the same technique for your nightly builds, feel free to use the code of snaps.php-tools.net as a starting point. You can get it from our subversion repository:`$ svn co http://www.php-tools.net/svn/patSnaps/trunk`Beware that some of the code is several years old and while it works with PHP 5.1 it surely could be optimized. Examples for the `autopackage2.php` file are available in the subversion repositories for `patTemplate`, `patForms` and `patBBCode`.

Posted by schst in PHP, PAT at 11:23

Friday, August 18. 2006

PHP Design Patterns finished

Today I received a message from O'Reilly, that my book PHP Design Patterns is now going to print and is scheduled to be released in September 2006. The work of the last six months is finally bearing fruit and I will be able to spend my spare time on coding again instead of just writing. If you wondered, why patTemplate or any of my other open source projects did not show any progress, this announcement should answer your questions.

I put nearly all of my thoughts on software architecture, OO design and patterns into this book and thus it consists of 370 pages dealing with OO development in PHP 5.1, creational patterns, structural patterns, behavioural patterns as well as enterprise patterns and MVC architectures. Furthermore it covers SPL, Propel and patTemplate to provide some real-life examples of the patterns.

If you intend to buy the book, you should be warned that it has been written completely in German. If you still are interested, you will find more information on the book as well as the code example on the website or you can already pre-order it from Amazon.

Thanks go to gERD, luckec and Frank who supported me with their knowledge as well as Toby, who introduced me to O'Reilly.

Posted by schst in PHP at 20:07

Thursday, August 17, 2006

patForms Gettext Patch

Usually i18n and multi language support is not a big issue in web business. Even if web-sites seem to be global players, most of them get along with support for very few languages. patForms for example comes with support for English, French and German and therefore covers pretty much everything I ever wanted.

On the contrary non-French, non-English and non-Germans may blame patForms for lack of support for Danish, Swedish, Spanish, Russian etc. Rightly, I suppose. Unfortunately, the supported languages are within the form elements and rules. Hence it would mean to change a lot of files (classes) to simply add another language's messages. Knowing that we started to introduce support for external translation tables - stored in ini-files.

Another idea was to use the PHP Gettext extension which should be available on most servers. The two major advantages of this method are: (1) You simply don't have to bother about NLS (native language support) while you write some geek patForm rules or element. (2) Gettext is widely spread and comes with a bunch of tools to extract translatable strings and maintain translation tables. This makes it fairly simple to add support for additional languages like Spanish, Danish and even Gungan.

So far, patForms does not use Gettext. Still, I created a patch which works extremely well. I love to invite you to test this patch and and tell me what you think about patForms using Gettext. Please also tell me your opinion whether we should move to Gettext for the next release of patForms.

Download:

- Zip file: [patForms-Gettext.zip](#) 512 kByte
- Tar Bz2 archive: [patForms-Gettext.tar.bz2](#) 188 kByte

Checkout from our Subversion repository

```
$ svn co http://www.php-tools.net/svn/patForms/branches/gettext/
```

Links:

- [PHP Gettext](#)
- [GNU Gettext](#)
- Translation editor [PO Edit](#), [KBabel](#)

Posted by gERD in PHP at 20:10

Monday, August 7, 2006

Der Zauberlehrling - Sorcerer's Apprentice

Recently the German PHP Magazine published the August 2006 issue. The principal topic is AJAX and Web 2.0 technologies, including an article covering the PEAR package HTML_AJAX

This article's target is (1st) to give a brief introduction to AJAX and how things work from Javascript an PHP side in contrary. Also (2nd) I wanted to show that you actually don't have to muck around with a mess of Javascript (browser compatibility issues and such) because it's it is all done by the Joshua Eichorn's HTML_AJAX framework.

Unfortunately, four odd pages arn't enough to build up a complete application. Still, it tells you where to begin and how you can add Web 2.0 features to your site within minutes. Also there is one more crux: it's German only Still, I hope you'll enjoy the article.

Posted by gERD in PHP at 08:58

Sunday, August 6, 2006

patTemplate 3.1.0a2 released

I just released patTemplate 3.1.0a2. This will be the last alpha version for 3.1.0 and I hope to release a stable version during the next 6-8 weeks. The major changes since 3.0.1a1 include: You may now use variables in conditions for sub-templates! added an input filter that allows Smarty syntax for variable modifiers! It is now possible to read templates from a database using PEAR::DB! The full list of changes can be found in the changelog. You can download the package using or PEAR channel server.

Posted by schst in PHP at 16:47

Exploring PHP to be published soon

Exploring PHP, a book I've been working on in march, will finally be published. I have been working on this together with Frank, Christian, Sandro and Kore and the book features five articles on advanced PHP topics. All articles in the book have been written in German.

The article I contributed showcases how event-driven-development can be used to create component based, flexible applications that allow you to easily plug in new features without modifying the core application. The examples in the book use PEAR's Event_Dispatcher. Other articles cover unit testing with SimpleTest (by Frank), the generation of 3D images with PEAR's Image_3D (by Kore), migration to PHP 5 and refactoring (by Sandro) as well as news on PHP 6 (by Christian). The editor for the book is Markus Nix, he compiled all the articles. If you take a closer look at the cover, you might see that there are different names listed on the cover than those mentioned in the blog. This is because some of the original authors did not manage to deliver their articles in time and have been replaced.

The planned release date for the book is 08/17/2006, so keep your eyes open.

Posted by schst in PHP at 12:01