designritter, photocase.com

Secure scripts with Apache Suexec

PLEASANT DREAMS

For many admins, the security of a web application is more important than its performance. If you have a web server with multiple users, the Suexec module can help you avoid problems associated with globally writable directories. **BY OLIVER FROMMEL**

If you follow security mailing lists, you know that web applications can be security nightmares. The reason for this is the subject of heated debate. Are languages like PHP to blame? Do too many web developers simply lack the skills to write secure applications? Whatever the cause, one thing is certain: something's got to change.

A holistic approach that redevelops every web server component from scratch is highly unlikely. Instead, the protagonists of the Apache project are working on improving individual components; although most people agree that this is not a perfect solution, it is infinitely better than *no* solution.

If you install packages from online forums, you often find a line in the manual telling you to set up a world-writable directory. Even if you are uneasy about this world access, the conventional approach to Apache security offers few alternatives. Normally, the server executes all scripts with the same user ID (*www*, *www-admin*, or something similar). Making directories writable to this user ID has the same effect as making directories world writable. No matter whether the directories are writable for all or just for the web server ID, the result is the same: everyone can write to everyone else's directory.

Suexec Apache Module

One possible exit route from the vicious circle is the Suexec Apache module. Suexec executes CGI scripts under a configurable user ID. The Suexec module limits the intruder's access to system resources by allowing the script to execute in a more limited a security context.

Although Suexec used to be difficult to use, the module is becoming fairly easy [1] to configure since the introduction of Apache 2. Most Linux distributions already include the Suexec module with the Apache package. For example, in Fedora, the httpd package includes the *mod_suexec.so* module file and the */usr/sbin/suexec* wrapper. In Debian Etch, you'll find Suexec in the package called *apache2*.

Launch the Apache server with the *-V* option to find out whether your Apache supports Suexec:

Listing 1: suexec -V

```
01 # /usr/lib/apache2/suexec -V
02  -D AP_DOC_ROOT="/var/www"
03  -D AP_GID_MIN=100
04  -D AP_HTTPD_USER="www-data"
05  -D AP_LOG_EXEC="/var/log/
    apache2/suexec.log"
06  -D AP_SAFE_PATH="/usr/local/
    bin:/usr/bin:/bin"
07  -D AP_UID_MIN=100
08  -D AP_USERDIR_SUFFIX="public_
    html"
```

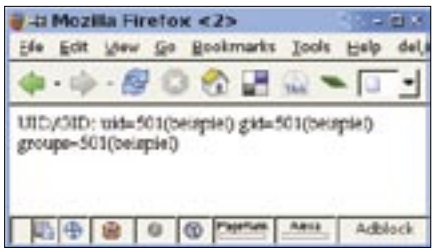

Figure 1: The CGI script will run with the user and group ID set in Suexec.

```
$ /usr/sbin/httpd ➥
-V | grep -i suexec
 -D SUEXEC_BIN= ➥
"/usr/sbin/suexec"
```

To provide the security most admins expect, a couple of pertinent values are built into the Suexec program, including the logfile, paths to executables, the document root, and the user and group IDs. Suexec has a command-line option that tells you how it is configured. Listing 1 shows these settings in Debian Etch.

The *DOC_ROOT* variable specifies the directory below which the user scripts must be stored. *SAFE_PATH* defines the path to the executables, which is passed to the scripts. Your Apache configuration needs to reflect this setting. In the example in this article, the document root directory */var/www* will contain multiple subdirectories, each of which belongs to a single user.

Simple Apache

Listing 2 shows the *suexec.conf* configuration file. Lines 6 through 8 set the required execute privileges for the user directory belonging to *example* in the Apache configuration; Line 3 sets this as the user's root directory. Line 10 defines files that end in *.cgi* as CGI scripts. If the Suexec module is loaded and the wrapper is available, all it takes to complete the configuration is Line 4 with its *SuexecUserGroup* directive.

Somebody who drops a CGI script into the */var/www/example* directory may get an internal server error or a blank page. A look at the Suexec logfile (for Fedora, */var/log/httpd/suexec.log*) tells you more:

```
target uid/gid (501/➥
501) mismatch with ➥
directory (501/501) ➥
 or program (500/501)
```

For security reasons, Suexec insists that directories and executables belong to the user listed in the Apache configuration. If the ownership details and privileges are correct, the following CGI script returns the desired results, as shown in Figure 1.

```
#!/bin/sh
echo "Content-type: text/html"
echo
echo "UID/GID:" `id`
```

Fast

It's safe to assume that many website operators will be unhappy at the thought of working with last century's CGI technology just for security reasons. As an alternative, you can use the FastCGI interface (FCGI), which gives you the ability to integrate PHP scripts with the Suexec infrastructure.

The *Mod_fcgi* file specified by FastCGI's developers has now been replaced by the mostly compatible *Mod_fcgid* (with a "d" at the end) [2], which is available as a binary module for most distributions.

The main issue with integrating FCGI and Suexec is getting the restrictive defaults of the two modules to match. If you use the system global PHP binary as the FCGI wrapper, Suexec will complain because it is located outside the document root. The solution is to copy it into a directory below the document root directory. The Apache configuration section looks like this:

```
AddHandler fcgid-script .php
FCGIWrapper ➥
/var/www/example/bin/php-cgi.php
```

This assumes an otherwise-working FCGI installation (typically a question of installing the package) and the CGI version of the PHP interpreter, which is also fine with FastCGI. Just call *php-cgi -v*.

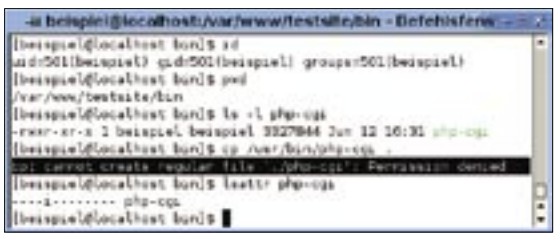

Figure 2: If the administrator sets the immutable flag for the PHP interpreter, not even the owner of the file can change it.

If you have been following this discussion attentively, you may have already noticed a slight problem: a web user could theoretically replace the PHP binary. If you are a provider running a server, you will want to avoid this for security reasons.

The solution is the immutable flag, which prohibits changes to a file even by the file's owner (Figure 2). The following example sets the flag for the user *example*'s PHP interpreter:

```
chattr +i /var/www/example/➥
bin/php-cgi
```

Thanks to this change, the Suexec/FCGI setup works as the doctor ordered, but only on Ext2/3, XFS, JFS, and Reiser filesystems, which have the immutable flag.

This admittedly complicated approach gives you the ability to run PHP scripts under the scrutiny of Suexec with a non-privileged user's rights. This configuration is at least a step in the right direction toward hardening a web server that hosts sites for multiple users.

Thanks to the flexibility of the FCGI interface, you can secure scripts in other programming languages, such as Ruby on Rails. PHP users may find that the SuPHP tool [3] is a useful alternative to the approach described in this article. ■

Listing 2: suexec.conf

```
01 <VirtualHost *>
02    ServerName example.example.
   com
03    DocumentRoot /var/www/
   example/
04    SuexecUserGroup example
   example
05
06    <Directory /var/www/
   example>
07       Options +ExecCGI
08    </Directory>
09
10    AddHandler cgi-script .cgi
11 </VirtualHost>
```

INFO

[1] Apache Suexec: *http://httpd.apache.org/docs/2.0/suexec.html*

[2] Mod_fcgid: *http://fastcgi.coremail.cn*

[3] SuPHP: *http://www.suphp.org*