suEXEC Support

The **suEXEC** feature provides users of the Apache HTTP Server the ability to run **CGI** and **SSI** programs under user IDs different from the user ID of the calling web server. Normally, when a CGI or SSI program executes, it runs as the same user who is running the web server.

Used properly, this feature can reduce considerably the security risks involved with allowing users to develop and run private CGI or SSI programs. However, if suEXEC is improperly configured, it can cause any number of problems and possibly create new holes in your computer's security. If you aren't familiar with managing *setuid root* programs and the security issues they present, we highly recommend that you not consider using suEXEC.

Before we begin

Before jumping head-first into this document, you should be aware that certain assumptions are made about you and the environment in which you will be using suexec.

First, it is assumed that you are using a UNIX derivative operating system that is capable of **setuid** and **setgid** operations. All command examples are given in this regard. Other platforms, if they are capable of supporting suEXEC, may differ in their configuration.

Second, it is assumed you are familiar with some basic concepts of your computer's security and its administration. This involves an understanding of **setuid/setgid** operations and the various effects they may have on your system and its level of security.

Third, it is assumed that you are using an **unmodified** version of suEXEC code. All code for suEXEC has been carefully scrutinized and tested by the developers as well as numerous beta testers. Every precaution has been taken to ensure a simple yet solidly safe base of code. Altering this code can cause unexpected problems and new security risks. It is **highly** recommended you not alter the suEXEC code unless you are well versed in the particulars of security programming and are willing to share your work with the Apache HTTP Server development team for consideration.

Fourth, and last, it has been the decision of the Apache HTTP Server development team to **NOT** make suEXEC part of the default installation of Apache httpd. To this end, suEXEC configuration requires of the administrator careful attention to details. After due consideration has been given to the various settings for suEXEC, the administrator may install suEXEC through normal installation methods. The values for these settings need to be carefully determined and specified by the administrator to properly maintain system security during the use of suEXEC functionality. It is through this detailed process that we hope to limit suEXEC installation only to those who are careful and determined enough to use it.

Still with us? Yes? Good. Let's move on!

suEXEC Security Model

Before we begin configuring and installing suEXEC, we will first discuss the security model you are about to implement. By doing so, you may better understand what exactly is going on inside suEXEC and what precautions are taken to ensure your system's security.

suEXEC is based on a setuid "wrapper" program that is called by the main Apache HTTP Server. This wrapper is called when an HTTP request is made for

a CGI or SSI program that the administrator has designated to run as a userid other than that of the main server. When such a request is made, Apache httpd provides the suEXEC wrapper with the program's name and the user and group IDs under which the program is to execute.

The wrapper then employs the following process to determine success or failure -- if any one of these conditions fail, the program logs the failure and exits with an error, otherwise it will continue:

1. **Is the user executing this wrapper a valid user of this system?**

   This is to ensure that the user executing the wrapper is truly a user of the system.

2. **Was the wrapper called with the proper number of arguments?**

   The wrapper will only execute if it is given the proper number of arguments. The proper argument format is known to the Apache HTTP Server. If the wrapper is not receiving the proper number of arguments, it is either being hacked, or there is something wrong with the suEXEC portion of your Apache httpd binary.

3. **Is this valid user allowed to run the wrapper?**

   Is this user the user allowed to run this wrapper? Only one user (the Apache user) is allowed to execute this program.

4. **Does the target CGI or SSI program have an unsafe hierarchical reference?**

   Does the target CGI or SSI program's path contain a leading '/' or have a '..' backreference? These are not allowed; the target CGI/SSI program must reside within suEXEC's document root (see `--with-suexec-docroot=`*`DIR`* below).

5. **Is the target user name valid?**

   Does the target user exist?

6. **Is the target group name valid?**

   Does the target group exist?

7. **Is the target user *NOT* superuser?**

   suEXEC does not allow *`root`* to execute CGI/SSI programs.

8. **Is the target userid *ABOVE* the minimum ID number?**

   The minimum user ID number is specified during configuration. This allows you to set the lowest possible userid that will be allowed to execute CGI/SSI programs. This is useful to block out "system" accounts.

9. **Is the target group *NOT* the superuser group?**

   Presently, suEXEC does not allow the *`root`* group to execute CGI/SSI programs.

10. **Is the target groupid *ABOVE* the minimum ID number?**

    The minimum group ID number is specified during configuration. This allows you to set the lowest possible groupid that will be allowed to execute CGI/SSI programs. This is useful to block out "system" groups.

11. **Can the wrapper successfully become the target user and group?**

Here is where the program becomes the target user and group via setuid and setgid calls. The group access list is also initialized with all of the groups of which the user is a member.

12. **Can we change directory to the one in which the target CGI/SSI program resides?**

    If it doesn't exist, it can't very well contain files. If we can't change directory to it, it might as well not exist.

13. **Is the directory within the httpd webspace?**

    If the request is for a regular portion of the server, is the requested directory within suEXEC's document root? If the request is for a `UserDir`, is the requested directory within the directory configured as suEXEC's userdir (see suEXEC's configuration options)?

14. **Is the directory *NOT* writable by anyone else?**

    We don't want to open up the directory to others; only the owner user may be able to alter this directories contents.

15. **Does the target CGI/SSI program exist?**

    If it doesn't exists, it can't very well be executed.

16. **Is the target CGI/SSI program *NOT* writable by anyone else?**

    We don't want to give anyone other than the owner the ability to change the CGI/SSI program.

17. **Is the target CGI/SSI program *NOT* setuid or setgid?**

    We do not want to execute programs that will then change our UID/GID again.

18. **Is the target user/group the same as the program's user/group?**

    Is the user the owner of the file?

19. **Can we successfully clean the process environment to ensure safe operations?**

    suEXEC cleans the process's environment by establishing a safe execution PATH (defined during configuration), as well as only passing through those variables whose names are listed in the safe environment list (also created during configuration).

20. **Can we successfully become the target CGI/SSI program and execute?**

    Here is where suEXEC ends and the target CGI/SSI program begins.

This is the standard operation of the suEXEC wrapper's security model. It is somewhat stringent and can impose new limitations and guidelines for CGI/SSI design, but it was developed carefully step-by-step with security in mind.

For more information as to how this security model can limit your possibilities in regards to server configuration, as well as what security risks can be avoided with a proper suEXEC setup, see the "Beware the Jabberwock" section of this document.

Configuring & Installing suEXEC

Here's where we begin the fun.

suEXEC configuration options

--enable-suexec
: This option enables the suEXEC feature which is never installed or activated by default. At least one `--with-suexec-xxxxx` option has to be provided together with the `--enable-suexec` option to let APACI accept your request for using the suEXEC feature.

--enable-suexec-capabilities
: **Linux specific:** Normally, the `suexec` binary is installed "setuid/setgid root", which allows it to run with the full privileges of the root user. If this option is used, the `suexec` binary will instead be installed with only the setuid/setgid "capability" bits set, which is the subset of full root privileges required for suexec operation. Note that the `suexec` binary may not be able to write to a log file in this mode; it is recommended that the `--with-suexec-syslog --without-suexec-logfile` options are used in conjunction with this mode, so that syslog logging is used instead.

--with-suexec-bin=*PATH*
: The path to the `suexec` binary must be hard-coded in the server for security reasons. Use this option to override the default path. *e.g.* `--with-suexec-bin=/usr/sbin/suexec`

--with-suexec-caller=*UID*
: The username under which httpd normally runs. This is the only user allowed to execute the suEXEC wrapper.

--with-suexec-userdir=*DIR*
: Define to be the subdirectory under users' home directories where suEXEC access should be allowed. All executables under this directory will be executable by suEXEC as the user so they should be "safe" programs. If you are using a "simple" `UserDir` directive (ie. one without a "*" in it) this should be set to the same value. suEXEC will not work properly in cases where the `UserDir` directive points to a location that is not the same as the user's home directory as referenced in the `passwd` file. Default value is "`public_html`".
  If you have virtual hosts with a different `UserDir` for each, you will need to define them to all reside in one parent directory; then name that parent directory here. **If this is not defined properly, "~userdir" cgi requests will not work!**

--with-suexec-docroot=*DIR*
: Define as the DocumentRoot set for httpd. This will be the only hierarchy (aside from `UserDir`s) that can be used for suEXEC behavior. The default directory is the `--datadir` value with the suffix "`/htdocs`", *e.g.* if you configure with "`--datadir=/home/apache`" the directory "`/home/apache/htdocs`" is used as document root for the suEXEC wrapper.

--with-suexec-uidmin=*UID*
: Define this as the lowest UID allowed to be a target user for suEXEC. For most systems, 500 or 100 is common. Default value is 100.

--with-suexec-gidmin=*GID*
: Define this as the lowest GID allowed to be a target group for suEXEC. For most systems, 100 is common and therefore used as default value.

--with-suexec-logfile=*FILE*
: This defines the filename to which all suEXEC transactions and errors are logged (useful for auditing and debugging purposes). By default the logfile is named "`suexec_log`" and located in your standard logfile directory (`--logfiledir`).

--with-suexec-syslog
: If defined, suexec will log notices and errors to syslog instead of a logfile. This option must be combined with `--without-suexec-logfile`.

--with-suexec-safepath=*PATH*

Define a safe PATH environment to pass to CGI executables. Default value is "`/usr/local/bin:/usr/bin:/bin`".

Compiling and installing the suEXEC wrapper

If you have enabled the suEXEC feature with the `--enable-suexec` option the `suexec` binary (together with httpd itself) is automatically built if you execute the `make` command.

After all components have been built you can execute the command `make install` to install them. The binary image `suexec` is installed in the directory defined by the `--sbindir` option. The default location is "/usr/local/apache2/bin/suexec".

Please note that you need ***root privileges*** for the installation step. In order for the wrapper to set the user ID, it must be installed as owner *`root`* and must have the setuserid execution bit set for file modes.

Setting paranoid permissions

Although the suEXEC wrapper will check to ensure that its caller is the correct user as specified with the `--with-suexec-caller` `configure` option, there is always the possibility that a system or library call suEXEC uses before this check may be exploitable on your system. To counter this, and because it is best-practise in general, you should use filesystem permissions to ensure that only the group httpd runs as may execute suEXEC.

If for example, your web server is configured to run as:

```
User www
Group webgroup
```

and `suexec` is installed at "/usr/local/apache2/bin/suexec", you should run:

```
chgrp webgroup /usr/local/apache2/bin/suexec
chmod 4750 /usr/local/apache2/bin/suexec
```

This will ensure that only the group httpd runs as can even execute the suEXEC wrapper.

Enabling & Disabling suEXEC

Upon startup of httpd, it looks for the file `suexec` in the directory defined by the `--sbindir` option (default is "/usr/local/apache/sbin/suexec"). If httpd finds a properly configured suEXEC wrapper, it will print the following message to the error log:

```
[notice] suEXEC mechanism enabled (wrapper:
/path/to/suexec)
```

If you don't see this message at server startup, the server is most likely not finding the wrapper program where it expects it, or the executable is not installed *setuid root*.

If you want to enable the suEXEC mechanism for the first time and an Apache HTTP Server is already running you must kill and restart httpd. Restarting it with a simple HUP or USR1 signal will not be enough.

If you want to disable suEXEC you should kill and restart httpd after you have removed the `suexec` file.

Using suEXEC

Requests for CGI programs will call the suEXEC wrapper only if they are for a virtual host containing a SuexecUserGroup directive or if they are processed by mod_userdir.

Virtual Hosts:
One way to use the suEXEC wrapper is through the SuexecUserGroup directive in VirtualHost definitions. By setting this directive to values different from the main server user ID, all requests for CGI resources will be executed as the *User* and *Group* defined for that <VirtualHost>. If this directive is not specified for a <VirtualHost> then the main server userid is assumed.

User directories:
Requests that are processed by mod_userdir will call the suEXEC wrapper to execute CGI programs under the userid of the requested user directory. The only requirement needed for this feature to work is for CGI execution to be enabled for the user and that the script must meet the scrutiny of the security checks above. See also the `--with-suexec-userdir` compile time option.

Debugging suEXEC

The suEXEC wrapper will write log information to the file defined with the `--with-suexec-logfile` option as indicated above, or to syslog if `--with-suexec-syslog` is used. If you feel you have configured and installed the wrapper properly, have a look at the log and the error_log for the server to see where you may have gone astray. The output of `"suexec -V"` will show the options used to compile suexec, if using a binary distribution.

Beware the Jabberwock: Warnings & Examples

NOTE! This section may not be complete.

There are a few points of interest regarding the wrapper that can cause limitations on server setup. Please review these before submitting any "bugs" regarding suEXEC.

suEXEC Points Of Interest

- Hierarchy limitations

  For security and efficiency reasons, all suEXEC requests must remain within either a top-level document root for virtual host requests, or one top-level personal document root for userdir requests. For example, if you have four VirtualHosts configured, you would need to structure all of your VHosts' document roots off of one main httpd document hierarchy to take advantage of suEXEC for VirtualHosts. (Example forthcoming.)

- suEXEC's PATH environment variable

  This can be a dangerous thing to change. Make certain every path you include in this define is a **trusted** directory. You don't want to open people up to having someone from across the world running a trojan horse on them.

- Altering the suEXEC code

  Again, this can cause **Big Trouble** if you try this without knowing what you are doing. Stay away from it if at all possible.