Symlink Race Condition Protection

Created by Unknown User (shavaun), last modified by Doc User on Nov 14, 2016

Overview

This document explains several options that you can use to implement symlink race condition protection. It also answers questions that many of our customers have asked about the patch that we provide through EasyApache.

Symlink race condition vulnerability

If you enable both of the `SymLinksIfOwnerMatch` and `FollowSymLinks` configuration settings, Apache becomes vulnerable to a race condition through symlinks. This symlink vulnerability allows a malicious user to serve files from anywhere on a server that strict OS-level permissions do not protect.

Summary of current options to address this issue

Filesystem-level solutions (best choices)

mod_ruid + jailshell: RECOMMENDED

Links

Tweak Settings - Security

Upside

To enable this option, recompile Apache and then enable *EXPERIMENTAL: Jailshell Virtual Hosts using mod_ruid2 and cPanel jailshell* in WHM's *Tweak Settings* interface (*Home >> Server Configuration >> Tweak Settings).*

Important:
This option is **not** available if you compile Apache with the MPM ITK module. You **cannot** use jailed shells with Apache's MPM ITK module.

Downside

Requires `mod_ruid2,` which is **EXPERIMENTAL**.

cagefs: RECOMMENDED

Links

CloudLinux Documentation

Upside

This option is available on all cPanel-supported platforms.

Downside

- Costs a nominal fee (requires CloudLinux™)
- Requires that you run the `cagefsctl --update` command after you make changes.

Kernel and Apache solutions

GRSec kernel patch

Links

grsecurity forums - View topic - Prevent Symlink Attack

Upside

This option provides kernel-level protection.

Downside

- Requires a custom kernel.
- Additional installation and maintenance burden.

mod_hostinglimits securelinks with CloudLinux kernel

Links

- Introducing SecureLinks for Apache
- CloudLinux Documentation

Upside

If you currently use CloudLinux, this option is already installed.

Downside

The directive will not affect VirtualHosts that do not have a specified user ID.

Apache-level patches (last resort choices):

Warning:
The following options are **not** recommended, because experienced malicious users can circumvent them. You should only use **one** of these options as a last resort if you cannot implement any of the above options.

Bluehost.com-provided patch (available via EasyApache):

Links

See Below

Upside

You can install this option via EasyApache.

Downside

- Protection from this patch is not as good as a kernel-level or a filesystem-level solution.
- This patch may slow the performance of high-traffic servers.
- Incompatible with Mailman.
- Incompatible with CGI Center apps.

Rack911-provided patch

Links

- http://layer1.rack911.com/before_apache_make
- cPanel Forums Post

Upside

This option is faster than the patch that EasyApache provides.

Downside

Protection from this patch is not as effective as a kernel-level or filesystem-level solution.

How to apply the symlink race condition patch available via EasyApache.

To help solve this issue, cPanel & WHM offers the option to apply a third-party patch (Bluehost.com) to Apache 2.X that will prevent the race condition.

To apply the patch, select *Symlink Race Condition Protection* from the *Exhaustive Options List* stage of the EasyApache interface.

Warning:
- By default, EasyApache does **not** apply this patch.
- This patch may slow the performance of high-traffic servers.
- If you already use a custom patch for the race condition (for example: `FollowSymLinks_to_OwnerMatch.patch`), you must either remove your custom patch or not enable EasyApache's *Symlink Race Condition Protection* option.

Frequently Asked Questions

What is the `FollowSymlinks` option in Apache?

This option allows Apache to serve files which are located outside of the virtual host's document root. (For example: `ln -s /home/bob/public_html/original.txt /home/bob/public_html/link.txt`).

What is the `SymLinksIfOwnerMatch` option in Apache?

If you enable symlinks via `FollowSymlinks`, Apache will only serve content via a symlink when the owner of the symlink matches the owner of the file (For example, the user `bob` links to a file owned by `bob`). However, Apache's documentation for symlinks states that there is a potential race condition with symlinks and the directive will affect server performance.

What is the race condition?

Apache performs a test for symlinks as it turns a URL into a file path on disk. Apache handlers expect to operate on the file path rather than on an open file handle. So, there is a race condition titled *time of check time of use* (TOCTOU) between the step where Apache validates the path and when a content handler accesses the symlinked file.

What does the Rack911 patch do?

The Rack911 patch turns the `FollowSymlinks` directive into '`FollowSymlinks`' + '`SymLinksIfOwnerMatch`'. This provides a quick way to activate both of these directives in `httpd.conf` at the same time.

How does the Rack911 patch protect my website from the race condition?

The Rack911 patch itself does not protect against the race condition. Instead, this patch changes `Options FollowSymlinks` into a shortcut for `Options FollowSymlinks SymLinksIfOwnerMatch`. If you use this patch, your users cannot turn off both`SymLinksIfOwnerMatch` and symlinks. Because Apache does not provide a detailed ACL approach for these options, this solution provides a quick way to enable and disable symlinks at a system-wide level within `httpd.conf`.

What does the new Bluehost.com patch do?

The Bluehost.com patch modifies Apache and APR so that Apache cannot access certain files. However, the Bluehost.com patch only affects requests that are processed through Apache's default handler (for example: static files such as `.html`, images, etc). The Bluehost patch does not affect requests that are processed by application content handlers (for example, `mod_php`, `mod_ruid2`, `mod_cgi`, `mod_suphp`, etc).

Traditionally, systems administrators assign content handlers to the content that they want to serve. For example, `mod_php` serves PHP scripts, and `mod_cgi` serves arbitrary CGI scripts. When Apache cannot find a content handler associated with an incoming request, Apache resorts to the `default_handler`. The `default_handler` reads in the contents of a file and displays it as-is to the requestor (for example, `cat file.txt > requestor`).

A problem occurs when an application runs as the `nobody` user (for example: `mod_php` configured with only the DSO handler). The application will create files on the server that the user `nobody` owns, which does not match the owner. Because the patch prevents access to static content that does not match the owner, the user would cannot view these files. A common example of this problem is an online PHP photo gallery.

Why would I want to use the Bluehost.com patch if the files it creates are inaccessible?

If your applications run as a specific user (for example: run `mod_php` with `mod_suphp`), then the files the application writes will have the correct ownership.

If the Bluehost.com patch prevents access to files that the user does not own, how does it determine what the user owns?

The Bluehost.com patch compares the UID of the requested file to the owner of the document root. If the owners do not match, Apache returns the `HTTP code 404; Not Found` error. Third-party modules may display additional errors (for example: `mod_ruid2` also emits an `Internal Server Error`).

Finally, the system writes a warning to the `error_log` that will resemble the following example:

```
[Tue Mar 19 18:08:31 2013] [error] [client 192.168.0.1] Caught race condition abuser. attacker: 529, victim: 0 open file owner: 0, open file:
```

How does CloudLinux handle the race condition?

CloudLinux uses the SecureLinks patch. This is a kernel-level patch that will not allow processes which run as specific groups (such as "nobody") to follow symlinks that a different user than the file or directory it links to owns.

For more information, read the CloudLinux documentation on SecureLinks.

How do the Bluehost.com patch and the Rack911 patch work together?

These two patches address the vulnerability in different ways. The Rack911 patch helps enforce access to certain files, while the Bluehost.com patch reduces the likelihood of the race condition. Unfortunately, we have received reports of conflicts from users who installed both patches.

Additional references

- http-dev mailing list thread for patch
- Syracuse University PDF on race condition
- Apache documentation on SymLinksIfOwnerMatch and FollowSymLinks configuration settings under Options

condition race symlink easyapache ea protection security apache