

Security - Keeping it Off the Web

Jamie

<http://www.geniegate.com/>

ABSTRACT

Describes a security concern affecting nearly every "off the shelf" PHP or CGI program, we'll uncover why the problem exists and what a website owner with very little technical background can do to protect themselves.

The topic of website security is seldom brought up among non-programmers and those who may not be technically inclined, yet if you operate a website, it is an issue of substantial concern to you.

Addressed in this article is a nearly universal problem, it affects almost every single PHP or CGI script I've ever seen. We won't delve into the technical issues too far, this is intended for the web site owner, someone who might install the occasional PHP file or CGI script. I will assume you are not a software developer.

A general rule I like to follow when installing php scripts or web applications is this; *If it doesn't need to be on the web, it shouldn't be there.* This is obvious, but it has implications that are not always clear.

What we are mainly referring to is configuration and to a lesser extent, program libraries and source code. It may also apply to files and other resources that are controlled through a script interface. An example of this would be scripts that charge money for downloading files or set up newsletters.

Lets start with an example, we'll call it *program.php*. In our example, *program.php* is a database application using mysql to store information.

For our script to do it's job, it will need access to your mysql password and user-name. It may also need access to many other program files and so-forth.

During installation, a control panel probably asked for your mysql credentials, it may even have asked you to change the file permissions on a configuration file of some sort. You've probably been through this type of install process at one time or another.

What it will do next, is write your database password and other private information to a configuration file. This usually happens without your knowledge, it is also where our problems begin.

Most people don't catch this right away, if the configuration file is in the same directory (or sub-directory) it is *web accessible*. Quite often it is a php file, usually with write permissions turned on.

The extension *.php* does afford some degree of protection, under normal circumstances these files aren't sent to a visitors browser but it is still unsafe.

If someone makes a slight mistake in the configuration or *.htaccess* file, it will dump the **actual contents** of "*conf.php*" to the users web browser, complete with your **database password** and other private information.

As anyone who has been around web servers very long can tell you, this is a common occurrence. I've personally seen it happen on several occasions.

Furthermore, many other web editing tools need to create backup files, resulting in something like *config.php.BAK* or perhaps *config.php.tmp*.

We now have a file ripe for hackers and other would-be intruders to gain access to your mysql database passwords as well as any other private information kept there.

Why is this so Common?

What is most alarming is that almost every single off the shelf web based program exhibits this very problem in some form or another.

You may ask yourself why a competent software designer would do such a thing? Why is it so common?

The answer is simplicity and market value. Customers demand software that is easy to install. It is actually done this way for your convenience.

Scripts that store everything in one directory make it easy for you, the website owner, to install and manage. Web software customers have come to demand this sort of thing.

Most software designers are aware of this problem, this is why they often choose to write configuration into a .php file, it gives you some protection at the expense of introducing other potentially security problems we won't cover here.

While java servlet technology provides WEB-INF/ (a secure place to store this type of information) most PHP or CGI environments do not offer such an environment.

Furthermore some web servers use something called "safe mode" which is a misguided attempt to make PHP "secure", operating in this mode forces a software developer to write insecure programs in the manner I've just outlined.

Correcting the Problem

For those of you who have inspected your website, good job! you've probably discovered you do have this problem. The next question is "What can be done"?

The correct approach is to relocate your configuration files (and any other information that shouldn't be web accessible) to a place safely outside your document root (sometimes called your "web folder") after you have done this, you shouldn't be able to access the files with a web browser, not even with a password.

I generally like to use the HOME directory, but it doesn't matter as long as it's safely kept well away from the prying eyes of would-be hackers.

With many scripts, this is not very practical, particularly if you have already installed them. At this point, damage control may be your only option.

If you're using apache, you can give yourself some added protection in the form of an .htaccess file, this is still not not secure, but it's certainly better than nothing at all. In "safe" mode, this may be your only option.

There are also things you can do via the language itself, but I won't cover them here.

After you have done this, it is important to make a note to yourself about it. When backing up your website, you will need to include not only the web pages, but these other files that you now have safely tucked outside your web directory.

If it's not Repairable

If you are unable to solve the problem, (for example the web script won't allow you to locate these files elsewhere) Simply being aware of it can go a long way toward protecting your website from future attackers. Now that you know, you can keep an eye out for any backup files that may have been created by other tools.

Try not to be too tough on whoever wrote the program, chances are they chose to do it this way for the convenience of their users. I can attest from first hand experience, if you don't store configuration data this way, customers can become annoyed and rarely understand your reasoning.