YGN ETHICAL HACKER GROUP

Article

An Apache Trick to Mitigate Shell File Attack

By

Aung Khant
Nov 2008

Web Application Security does not greatly depend on software version or specific vendor but on its authors. Web Developers are always in rush to finish their projects in time. They definitely have to use freely available applications and scripts out there that do not check the validity of uploaded files from user. Users can upload any files to any directories, mixing some vulnerabilities like Directory Traversal. File upload vulnerability is a commonly seen flaw that allows attackers to upload php shell files. These shell files can do anything depending on the security of web server. In web servers where insecure configuration exists, these shell files can bring the maximum benefit for attackers. To name a few, they can do php safemode bypassing, port scanning, database connectivity, remote backdoor port opening, ftp brute forcing, kernel exploiting and so on.

More and more features are being added to php shell files which are sold at a reasonable price at underground forum sites. Such commercial ones are kept private and all antivirus (AV) softwares cannot even detect them. Today's AV wares detect popular freely distributed shell files only if they have not been modified to some extent. So, your ClamAV installed on your web server or proxy is not of great help to you.

The solution to this is to disable dangerous php API functions which interact with underlying Operating System. Unluckily, users who use shared hosting do not have a chance to modify php.ini. The trick is to use .htaccess to disable dangerous php functions if web server admins have not overridden php settings. The format is:

[.htaccess of your web root dir]

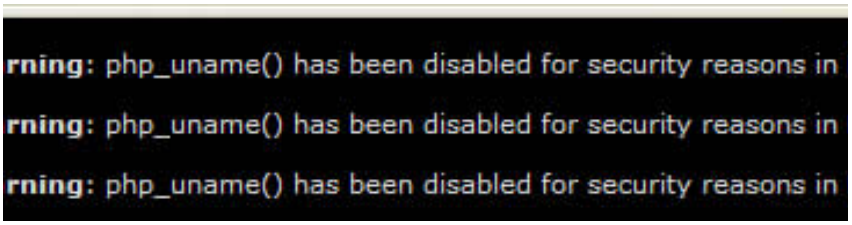
```
php_value disable_functions functions_list_separated_by_commas

php_value disable_functions shell_exec,system,exec,popen
```

[/.htaccess]

Which functions to disable depends on your needs. But at least you should disable shell_exec, system, exec, passthru, and popen. A comprehensive list is :

shell_exec,system,exec,passthru,popen,pclose,proc_close,proc_get_status,proc_nice,proc_open,proc_terminate,link,dl,apc_clear_cache,apc_store,apc_fetch,apc_delete,apc_define_constants,apc_load_constants,apc_compile_file,get_defined_functions, apache_get_modules,apache_get_version,apache_getenv,apache_setenv,fsockopen,highlight_file,ini_alter,ini_restore,phpinfo,show_source,symlink,rmdir,mkdir,php_uname,getcwd,unlink,glob,readdir

Although this method does not provide 100% security, you will see shell file will probably face memory exhaustion when you run it. It cannot do much as shell author expects it to do. This kicks off script kiddies.


rning: php_uname() has been disabled for security reasons in
rning: php_uname() has been disabled for security reasons in
rning: php_uname() has been disabled for security reasons in


r: Allowed memory size of 268435456 bytes exhausted (tried to allocate 63 bytes)
on line 2294