metaparadigm

PHP {in}security

Michael Clark <michael at metaparadigm dot com>

Presentation Overview

- PHP {in}security
- php.ini
- FastCGI
- suEXEC
- chroot jail
- readonly || noexec
- mod_security2
- Conclusion

PHP {in}security

- Many vulnerabilities in exist common PHP software packages
- How do we protect against these vulnerabilities
- Need to provide an environment with minimal trust for PHP applications
- Only allow a minimal set of essential permissions in a hardened environment
- Need to use privilege separation and chroot jails

php.ini

- **`open_basedir = /var/www/mysite`**
  - Only allow open to essential directories.
- **`session.save_path = /var/tmp/php-sessions`**
  - Don't allow scripts to read sessions and isolate from /tmp.
- **`upload_tmp_dir = /var/www/mysite/tmp`**
  - Isolate upload tmp files from /tmp.
- **`display_errors = 0`**
  `expose_php = 0`
  - Avoid information leaks.
- **`allow_url_fopen = 0`**
  `allow_url_include = 0`
  - Avoid commonly exploited vulnerabilities.

FastCGI

- **mod_fcgi or mod_fastcgi?**
  - mod_fcgi was adopted as an official Apache project.
- **Process Separation**
  - PHP no longer runs inside of the webserver as with default Apache and mod_php.
  - Allows more control over PHP when it is running as a separate process.
- **Apache Worker MPM**
  - Threaded MPM in Apache 2.x can't be used with mod_php as PHP is not fully thread safe.
  - By-product of FastCGI is that worker MPM can now be used.
  - Ability for Apache to handle more connections.

suEXEC

- **Apache suEXEC augments mod_fcgi**
  - Allow CGI and FastCGI process to be started as a different user to the web server.
- **Privilege Separation**
  - Run PHP FastCGI process as a user that does not have write access to the web site directories.
  - Apache can update these directories e.g. mod_dav but PHP can no longer has write perms on these same directories.
- **More than one user per vhost**
  - Apache Hack – create special internal vhosts for each user and then use proxy rewrite rules.
    - RewriteRule /cgi-bin/(.*) http://cgi-bin-user/cgi-bin/$1 [P]

Chroot jail

- **Patch suEXEC to chroot**
  - Be very careful changing suEXEC as there are many security checks that must be observed.
  - Patch available from “Apache Security” author 'Ivan Ristic'
    - http://www.apachesecurity.net/tools/
- **Combine with FastCGI and bind mounts**
  - Create minimal chroot jail for PHP
    - only executable is php-cgi
    - Isolate from other services on the same box
  - Bind mount website directory

```
mount -o bind,noexec /var/www/mysite /var/phpchroot/var/www/mysite
```

readonly || noexec

- **Mount PHP chroot image readonly**
  - Readonly loop mount is a nice easy way to make PHP jail
  - *Can't write in dirs that can be exec'ed*
- **Mount and writables noexec**
  - Bind mount /var/www/mysite into jail with noexec
  - *Can't exec in dirs that can be written to*
- **No other interpreters in PHP chroot jail**
  - Can't “perl /tmp/foo.pl” or “sh /tmp/foo.sh”
- **Start with zero write access for PHP**
  - Add writable directories on an as needed basis
  - Dont include from any writable dirs

mod_security2

- **Web Application firewall for Apache**
- **Detects common exploits within Apache before being passed to PHP or other web apps:**
  - Avoids known weaknesses and vulnerabilities.
  - Detects special characters in URLs and inspects POST content type and payload.
  - Detects common injection patterns and exploits.
  - Adds additional logging (Full Request Headers).
  - Flexible extensible rules.

Conclusion

- **Many PHP applications available with unknown trust and security**
- **We can't always audit and fix all of them**
- **We can run them in a minimally trusted sandbox**
- **Default Apache + mod_php is not very secure**
  - Vulnerabilities in many well known PHP applications.
- **FastCGI + suEXEC + chroot + PHP + (readonly || noexec) + mod_security2**
  - Provides a minimally trusted environment for running untrusted PHP applications.
- **To do – SELinux, Hardened PHP, Suhosin**