REV 1.1 2010-Apr-11

PHP Application Security Checklist

BASIC

- ☐ Strong passwords are used.
- ☐ Passwords stored safely.
- ☐ register_globals is disabled.
- ☐ Magic quotes is disabled.
- ☐ Error reporting is disabled.
- ☐ Server(s) are physically secure.

INPUT

- ☐ Input from $_GET, $_POST, $_COOKIE, and $_REQUEST is considered tainted.
- ☐ Understand that only some values in $_SERVER and $_ENV are untainted.
- ☐ $_SERVER['PHP_SELF'] is escaped where used.
- ☐ Input data is validated.
- ☐ \0 (null) is discarded in input.
- ☐ Length of input is bounded.
- ☐ Email addresses are validated.
- ☐ Application is aware of small, very large, zero, and negative numbers. Sci. notation too.
- ☐ Application checks for invisible, look-alike, and combining characters.
- ☐ Unicode control characters stripped out when required.
- ☐ Outputted data is sanitized.
- ☐ User-inputted HTML is sanitized with HTMLPurifier.
- ☐ User-inputted CSS is sanitized using a white-list.
  - ☐ Abusable properties (position, margin, etc.) are handled.
  - ☐ CSS escape sequences are handled.
  - ☐ JavaScript in CSS is discarded (expressions, behaviors, bindings).
- ☐ URLs are sanitized and unknown and unwanted protocols are disallowed.
- ☐ Embedded plugins are restricted from executing JS.
- ☐ Embedded plugin files (Flash movies) are embedded in a manner so that only the intended plugin is loaded.
- ☐ The application uses a safe encoding.
  - ☐ An encoding is specified using a HTTP header.
  - ☐ Inputted data is verified to be valid for your selected encoding if using an unsafe encoding.

FILE UPLOADS

- ☐ Application verifies file type.
  - ☐ User-provided mime-type value is ignored.
  - ☐ Application analyzes the content of files to determine their type.
  - ☐ It is understood that a perfectly valid file can still contain arbitrary data.
- ☐ Application checks the file size of uploaded files.
  - ☐ MAX_FILE_SIZE is not depended upon.
  - ☐ File uploads cannot "overtake" available space.
- ☐ Content is checked for malicious content.
  - ☐ Application uses a malware scanner (if req.).
  - ☐ Uploaded HTML files are displayed securely.
- ☐ Uploaded files are not moved to a web-accessible directory.
- ☐ Extensive path checks are used when serving files.
- ☐ Uploaded files are not served with include().
- ☐ Uploaded files are served as an attachment using the Content-Disposition header.
- ☐ Application sends the X-Content-Type-Options: nosniff header.
- ☐ Files are not served as "application/octet-stream", "application/unknown", or "plain/text" unless necessary.

DATABASE

- ☐ Data inserted into the database is properly escaped or parameterized/prepared statements are used.
  - ☐ addslashes() is not used.
- ☐ Application does not have more privileges to the database than necessary.
- ☐ Remote connections to the database are disabled if they are unnecessary.

SERVING FILES

- ☐ User input is not directly used in a pathname.
  - ☐ Directory traversal is prevented.
  - ☐ Null (\0) in paths filtered.
  - ☐ Application is aware of '~'.
  - ☐ PHP streams are filtered.
- ☐ Access to files is not restricted by hiding the files.
- ☐ Remote files not included with include().

AUTHENTICATION

- ☐ Bad password throttling.
  - ☐ CAPTCHA is used.
- ☐ SSL used to prevent MITM.
- ☐ Passwords are not stored in a cookie.
- ☐ Passwords are hashed.
  - ☐ Per-user salts are used.
  - ☐ crypt() is used with sufficient number of rounds.
  - ☐ MD5 is not used.
- ☐ Users are warned about obvious password recovery questions.
- ☐ Account recovery forms do not reveal email existence.
- ☐ Pages that send emails are throttled.

SESSIONS

- ☐ Sessions only use cookies. (session.use_only_cookies)
- ☐ On logout, session data is destroyed.
- ☐ Session is recreated on authorization level change.
- ☐ Sites on the same server use different session storage dirs.

3RD-PARTIES

- ☐ CSRF issues are prevented with tokens/keys.
  - ☐ Referrers are not relied upon.
  - ☐ Pages that perform actions use POST.
  - ☐ Important pages (logout, etc.) are protected.
- ☐ Your pages are not written in a way (i.e. JSON, JS-like) where they can be included and read on a remote website successfully.
- ☐ Aware that Flash can bypass referrer checks to load images and sound files.
- ☐ The following things will not reveal significant information if included remotely:
  - ☐ Images.
  - ☐ Pages that take a longer time to load.
  - ☐ CSS files.
  - ☐ Existence or ordering of frames.
  - ☐ Existence of a JS variable.
  - ☐ Detected visit of a URL.
- ☐ Inclusion of your website in an inline frame with JS disabled does not reveal a threat.
- ☐ Application uses frame busting code and sends the X-Frame-Options header.

MISCELLANEOUS

- ☐ A cryptographically-secure PRNG is used for secret randomly-generated IDs (activation links, secret IDs, etc.).
  - ☐ Suhosin is installed or you are not using rand() or mt_rand() for this.
- ☐ Anything that consumes a lot of resources should be throttled and limited.
  - ☐ Pages that use 3rd-party APIs are throttled.
- ☐ You did not create your own encryption algorithm.
- ☐ Arguments to external programs (i.e. exec()) are validated.
- ☐ Generic internal and external redirect pages are secured.
- ☐ Precautions taken against the source code of your PHP pages being shown due to misconfiguration.
- ☐ Configuration and critical files are not in a web-accessible directory.

SHARED HOSTING

- ☐ Using a secure shared host where users cannot access the files of other users.
- ☐ Aware that fellow shared hosting users:
  - ☐ Can, if on the same IP address, issue requests against your site with XMLHttpRequest in IE6.
  - ☐ Can access your website from 127.0.0.1 or ::1.
  - ☐ Can host a server on the same IP address.
  - ☐ Are not "remote" as far as your DB is concerned.
- ☐ Session & file upload directories are not shared.

Find the annotated original at: http://sk89q.com/phpsec/