

# Re: [PHP] help create community newbie guide to security

*Source:* <http://coding.derkeiler.com/Archive/PHP/php.general/2003-11/0920.html>

---

*From:* Chris Shiflett (*shiflett\_at\_php.net*)

*Date:* 11/11/03

Date: Mon, 10 Nov 2003 20:21:06 -0800 (PST)

To: "Chris W. Parker" <cparker@swatgear.com>, php-general@lists.php.net

— "Chris W. Parker" <cparker@swatgear.com> wrote:

- > *What I'd like to do is gather enough info to be able to write a good,*
- > *short (heck in can be long, I don't care) write up on what it takes to*
- > *write a secure app and be able to post a link to said document any time*
- > *someone asks a question security related.*

It would probably need to be either very short or very long. If it is very short, I think people will safely assume that it is incomplete and only covers the most important topics. If it is very long, it can be more complete. Anything in between has the danger of giving people a false sense of security. You might want to choose which approach you plan to use from the beginning.

- > *I know there are some good minds on this list when it comes to security*
- > *so I'm hoping that these minds will contribute concise/verbose/detailed*
- > *information.*

I'm not sure what you consider to be a good mind, but I'd be happy to help. I'm in favor of any project that helps the PHP community. We all benefit from such things.

- > *1. Data that can be used to authenticate a user should never be stored*
- > *on the clients machine even if it is obfuscated, and/or "encrypted" in*
- > *some fashion.*

Nice point. Perhaps it would be better to mention why? As a developer, I sometimes have to be shown what bad stuff can happen before I really "get it". Even a casual mention of how much easier this can make impersonation would help a bit in driving the point home.

- > *2. The session id should not be stored on the client.*

I'm not sure I agree with this. Can you elaborate? Though it can be argued that sending the session identifier is a security risk, it is the one that is necessary for state (and therefore session) management.

- > 3. *A unique identifier should be created for the user when the user logs in. This unique identifier should not be based on the user's name, password, id, or session id.*

What is the difference between the session and unique identifiers?

- > 6. *The login page should be done over HTTPS. (Sounds like a good idea but is this necessary? Is this killing a mouse with a jackhammer?)*

It depends on the amount of risk that is acceptable. Requests for non-SSL URLs are sent in the clear and are vulnerable to things such as sniffing, man in the middle attacks, and other similar attacks.

Yahoo offers the user a choice between SSL and non-SSL logins, and they default to non-SSL. Some sites describe the choice as secure versus fast.

- > 7. *All input received via \$\_POST and/or \$\_GET should be thoroughly filtered*

I would say any data that originates from any external source should be treated as tainted data until it can be properly verified via data filtering. In addition, a white list approach is much safer than a black list approach in this regard.

- > *The best practice regarding filtering is to only allow what you want and reject the rest instead of the other way around (rejecting everything you don't want, accepting the rest).*

Oops, yeah, what you said. :-)

- > 8. *\$\_SESSION data is safe considering it can't be modified unless someone has access to your box which is an even bigger problem.*

This is a very good point, although it's hard to express. You don't want anyone to misinterpret what you are saying to mean that data is magically validated prior to being stored in the session. This is the developer's responsibility. This may seem obvious, but if this document is intended for the general public, more clarity is always good.

What is important is the point that you touch on; session data can't be directly modified by the end user like the data found in \$\_COOKIE, \$\_GET, and \$\_POST. It is server data, which is an important difference that can be leveraged by a careful developer.

- > 10. *Use htmlentities() on data that will be put through a SQL query to prevent XSS attacks. <http://php.net/htmlentities>*

This is a nice suggestion. While htmlentities() cannot be guaranteed to defend against all XSS vulnerabilities, I would bet that most XSS vulnerabilities are due to a complete lack of filtering logic. If a developer doesn't even bother using htmlentities(), neglect is the best

php.general: Re: [PHP] help create community newbie guide to security

word to describe his/her approach to developing.

In some cases, the developer may want certain HTML elements interpreted rather than escaped in this way. Perhaps you could mention that something like `str_replace()` can be used to convert specific HTML entities back to their original form. This method should filter any unwanted elements.

> *11. Some good links to more information are... (add some links here, but  
> let's not make this a link fest!)*

<http://www.php.net/manual/en/security.index.php>

Hope that helps.

Chris

=====

My Blog

<http://shiflett.org/>

HTTP Developer's Handbook

<http://httphandbook.org/>

RAMP Training Courses

<http://www.nyphp.org/ramp>